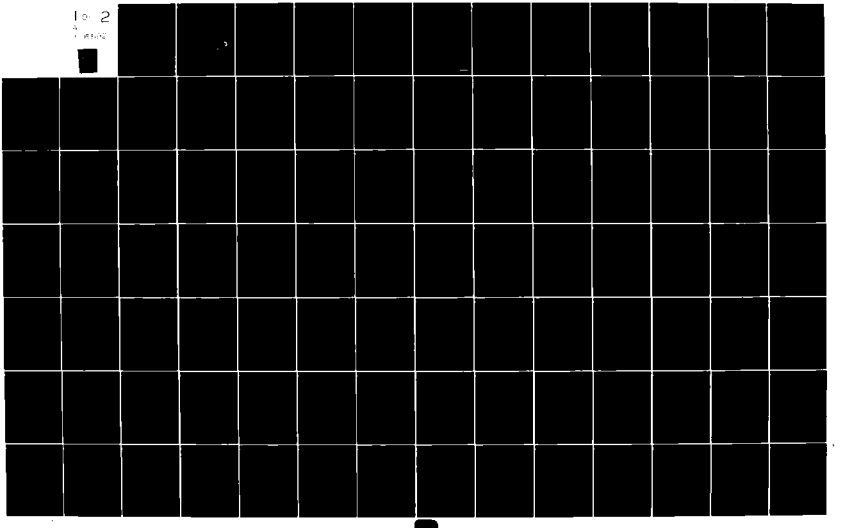
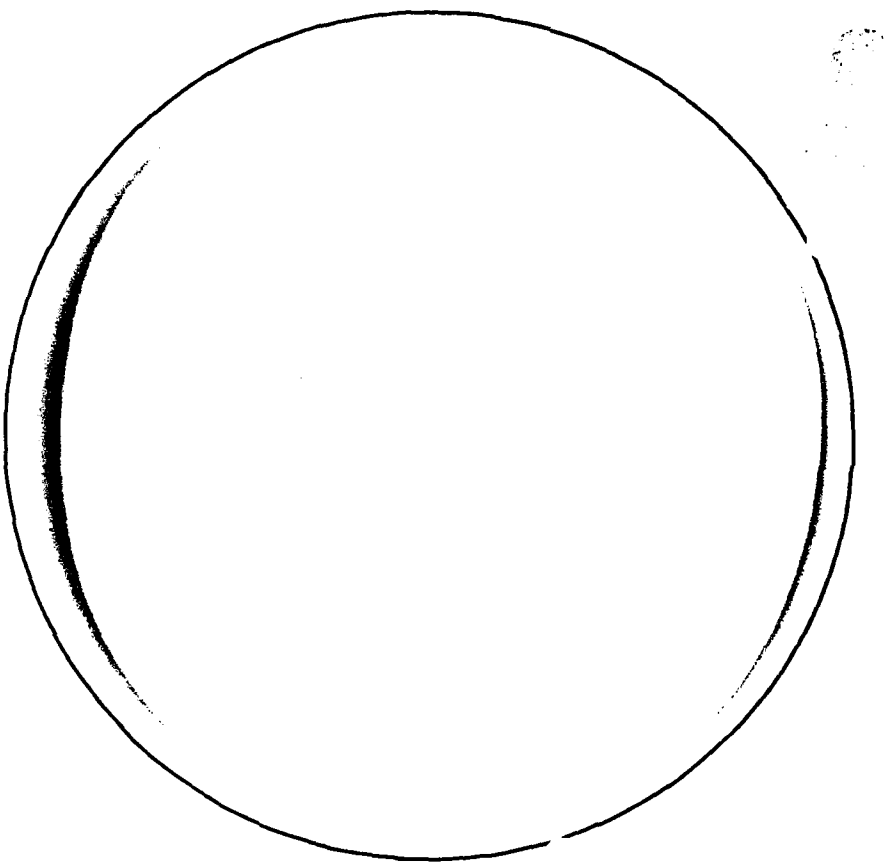
AD-A116 592 ALFRED P SLOAN SCHOOL OF MANAGEMENT CAMBRIDGE MA CEN--ETC F/G 9/2
SHELL: A SIMULATOR FOR THE SOFTWARE TEST VEHICLE OF THE INFOPLE--ETC(U)
JAN 82 T TO
UNCLASSIFIED CISR-M010-8201-08 N00039-81-C-0663
NL



(12)



Center for Information Systems Research

Massachusetts Institute of Technology
Sloan School of Management
77 Massachusetts Avenue
Cambridge, Massachusetts, 02139

83

Contract Number N00039-81-C-0663 (MIT # 91445)
Internal Report Number M010-8201-08
Deliverable Number 4

12

SHELL:
A SIMULATOR FOR THE SOFTWARE TEST VEHICLE OF THE INFOPLEX DATABASE COMPUTER

Technical Report #8

By
Tak To

January, 1982

Principal Investigator:
Professor Stuart E. Madnick

Prepared for:
Naval Electronics System Command
Washington, D.C.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: Technical Report #8	2. GOVT ACCESSION NO.: AD-A115 592
3. RECIPIENT'S CATALOG NUMBER:	
4. TITLE (and Subtitle): SHELL: A Simulator for the Software Test Vehicles of the INFOPLEX Database Computer	5. TYPE OF REPORT & PERIOD COVERED:
	6. PERFORMING ORG. REPORT NUMBER: M010-8201-08
7. AUTHOR(s): Tak To	8. CONTRACT OR GRANT NUMBER(s): N0039-81-C-0663
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Sloan School of Management, 50 Memorial Drive, Cambridge, MA 02139	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS:
11. CONTROLLING OFFICE NAME AND ADDRESS:	12. REPORT DATE: January 1982
	13. NUMBER OF PAGES: 161
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office):	15. SECURITY CLASS. (of this report): unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

database computer, database management system, Software Test Vehicle, distributed operating system

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The INFOPLEX database computer is a special computer designed for large scale information management. The information management functions are decomposed into a functional hierarchy implemented by a hierarchy of micro-processors. Decentralized control mechanisms are used to coordinate the activities of individual modules in the hierarchy.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Before realizing INFPLEX in hardware, it is essential to validate all the design details via a software test vehicle (STV). A simulator (SHELL) is built to provide the necessary facilities for the operating of this software test vehicle. It has two parts: an event simulator which simulates the operation of the afore mentioned hardware configuration; and an operating system emulator which provides the the environment for testing the multi-threading, parallel processing application programs. SHELL is meant to be used as the Control Structure portion of the STV project, which includes two additional parts, the Functional Hierarchy STV and the Storage Hierarchy STV.

SHELL:
A SIMULATOR FOR THE SOFTWARE TEST VEHICLE OF THE INFOPLEX DATABASE COMPUTER

by

Tak To

Submitted to the Department of Electrical Engineering and Computer Science, on May 22, 1981
in partial fulfillment of the requirements
for the degree of the Bachelor of Science

ABSTRACT

The INFOPLEX database computer is a special computer designed for large scale information management. The information management functions are decomposed into a functional hierarchy implemented by a hierarchy of micro-processors. Decentralized control mechanisms are used to coordinate the activities of individual modules in the hierarchy.

Before realizing INFOPLEX in hardware, it is essential to validate all the design details via a software test vehicle. A simulator (SHELL) is built to provide the necessary facilities for the operating of this software test vehicle. It has two parts: an event simulator which simulates the operation of the afore mentioned hardware configuration; and an operating system emulator which provides the environment for testing the multi-threading, parallel processing application programs.

SHELL facilitates the following goals: 1) the validation of the logic and algorithm of the application programs; 2) the validation of the multi-threaded, distributed control design of the functional hierarchy; 3) the investigation of detailed hardware designs (e.g., buffer sizes); and 4) the collection of performance data as a basis for further design considerations.

Table of Content

Table of Figures

Table of Tables

1.0 INTRODUCTION

1.1. INFOPLEX

The INFOPLEX database computer is a special purpose computer designed for large-scale information management. [Madnick] The specific objective of INFOPLEX includes providing substantial information management improvement over conventional architecture (e.g., up to 1000-fold increases in throughput) supporting very large complex databases (e.g., over 100 billion bytes of structured data), and providing extremely high reliability.

To achieve these goals, the architecture of INFOPLEX is built around the concept of hierarchical decomposition. The functions of information management are decomposed into a functional hierarchy. Each subfunction with a level of the hierarchy is implemented by means of a complex of processors. Highly parallel operations at each level and among levels boost the performance as well as the reliability. [Hsu 1]

A large capacity, cost-effective memory with rapid access time is realized using an 'intelligent' storage hierarchy. With a high degree of parallelism in operation, the storage hierarchy is able to support the storage requirements of the INFOPLEX functional hierarchy. The control of the storage hierarchy is distributed and micro-processors are used to implement

these control mechanisms.

1.2. Scope of the Current Project

Before realizing INFOPLEX in hardware, it is essential to validate all the design details via a software test vehicle. The current project is to design and implement a set of programs so as to provide the necessary facilities for the operation of this software test vehicle.

SHELL is a set of PL/1 programs on an IBM 370 VM/CMS system. It can be divided logically into two parts. The first is a special purpose event simulator which simulates the parallel operating environment in which a number of machines -- processors, gateway controllers etc. -- execute independently and compete for resources. Of specific interest is the communication mechanism among the processors. The interaction among the different machines along the data path is detailedly tracked.

The simulator can realistically reflect the actual operation of the hardware environment, hence the hardware design can be tested before it is actually built. In addition, performance data are gathered to serve as a basis for further detailed specifications (e.g., sizes of various buffers). Even though the main purpose is to test the functional hierarchy, some of the results can be applied to that of the storage hierarchy as well.

The second part of SHELL is an operating system emulator. It provides the parallel operating environment of a multi-processor, multi-process operating system, as well as the facilities of inter-process communication and inter-process synchronization. Application programs written in PL/1 can run directly under this emulated operating system.

Thus, the algorithms in the program modules of the functional hierarchy can be fully tested. In addition, since the simulator provides a realistic reflection of the actual execution speed, the relative efficiencies of the different algorithms and the different approaches of functional decomposition can be measured.

1.3. Structure of this Paper

Following this introduction, Chapter 2 outlines the hardware configuration of INFOPLEX and Chapter 3 describes how it is simulated by the simulator. Chapter 4 describes the functions of the local operating system in INFOPLEX, and Chapter 5 describes how they are emulated by the emulator. Chapter 6 describes the program organization of SHELL, its execution logic, and the data structures it uses. Chapter 7 presents the user-interface aspects of SHELL as a user's guide. A test program and a sample simulation session are included as examples. Chapter 8 is the conclusion.

2.0 THE HARDWARE ENVIRONMENT OF INFOPLEX

2.1. Overview

The hardware environment of the INFOPLEX datacomputer is shown in Figure 2.1. [Hsu 2] It is composed of multiple processors and memory modules. Processors and memory modules are grouped into clusters, each cluster corresponding to a level in either the storage heirarchy or the functional heirarchy. At each level, there is a common pended bus that joins all the hardware. For communication among levels, there is a global bus that connects to the local bus of each level via a gateway controller.

At each level, a number of processors sharing a large memory module operate a multi-processor operating system. (The local operating system of that level.) Each processor may have additional private RAMs, ROMs, or co-processors. The processors are homogeneous -- each processor executes the same operating system code to dispatch itself and there is no overall master supervisor processor.

Each data path connecting to the local pended bus (including that of the gateway controller) is buffered by a latch/buffer so that each device can access the bus asynchronously. It eliminates the necessity of special software buffering while waiting for a resource -- the shared memory or the gateway controller. The connection

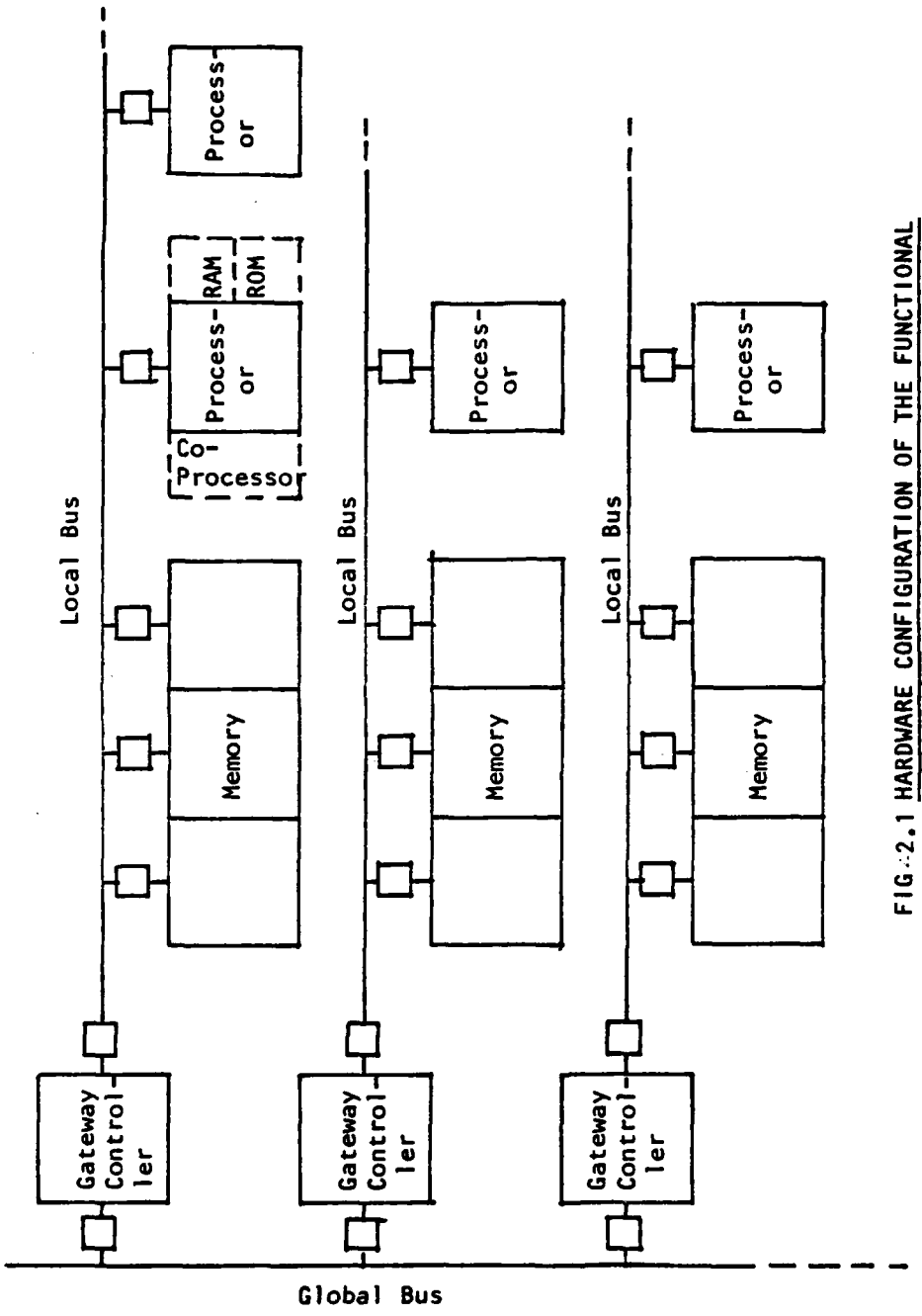


FIG. 2.1 HARDWARE CONFIGURATION OF THE FUNCTIONAL HIERARCHY OF INFOPLEX

between the global bus and the gateway controllers is similar in nature.

There is no shared memory between levels. Data are sent from the processor of one level, through the local gateway controller, the global bus, and the foreign gateway controller to the memory module of the destination level.

The data transfer from the processor to the gateway controller is memory-mapped. When a processor wants to transfer data to another level, it sends the data to a virtual address which will be recognized by the local gateway controller. (The destination level will either be coded in the address, or be a prefix in the data stream.) The gateway controller, serving as an entrepot between the local bus and the global bus, will send the data to the global bus.

The global bus is interleaved: data streams with different sources and destinations can share the global bus at the same time.

The gateway controller at the destination level will pick up the data stream from the global bus and deposit it (without the intervention of a processor) into the local shared memory module.

Because software buffering is not necessary along the path between the sending processor and the destination memory, the data transfer between two levels is

immediate in nature. To maintain a high throughput rate of interlevel data transfer, a special protocol is set up between a gateway controller and the local operating system.

2.2. GC-LOS Protocol

Two types of data transfer are recognized in this protocol: 'data blocks' and 'service requests.' Data blocks (or type 'D') are fixed sized, while service requests (or type 'S') are variable length data streams.

The local operating system maintains the following: a Data Request Queue (DRQ), a Service Request Queue (SRQ), and a Data Block Buffer (DBB).

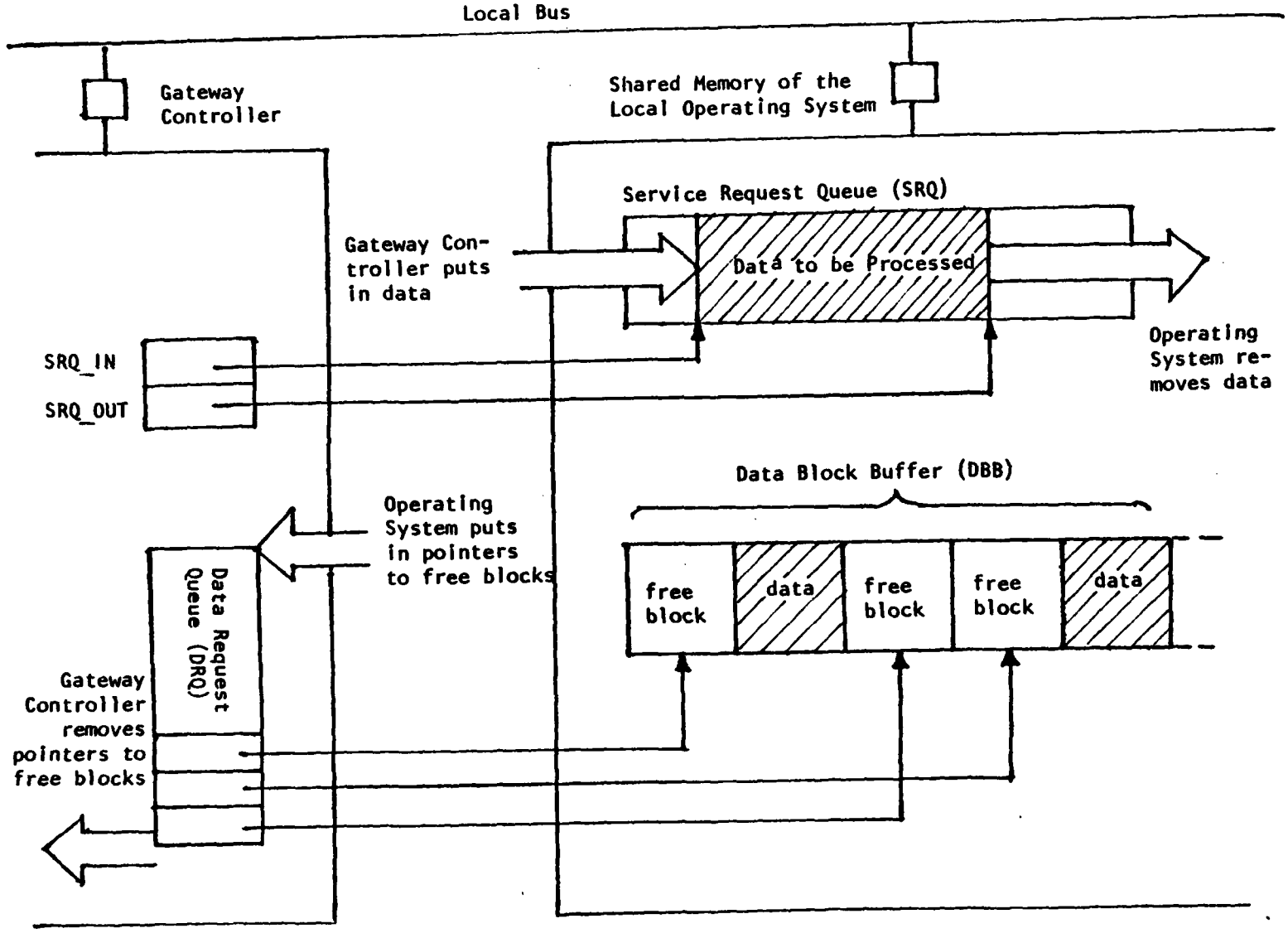
The SRQ is a software FIFO buffer (ring buffer) residing in the shared memory, with the port pointers SRQ_IN and SRQ_OUT residing in the gateway controller (as special registers, for example). The processors of the local operating system can access the pointers directly. SRQ_IN points to the next available location in the SRQ for storing incoming S-type data, and SRQ_OUT points to the first location of the next type S-type data stream for the operating system to process. SRQ_OUT chases SRQ_IN as S-type data are put in and processed. The local operating system is responsible for processing the data (or at least removing it from the SRQ) fast enough, and for maintaining a reasonable size of empty space in the SRQ.

The DRQ is a FIFO buffer (either software or hardware) residing in the gateway controller. The DRQ contains pointers to the free blocks in the DBB into where the gateway controller can put incoming data blocks (type 'D'). When a D-type data block arrives, the gateway controller obtains a pointer from the DRQ and put the data block into the free block in DBB pointed to by the pointer. The local operating system is responsible for maintaining a reasonable number of free blocks in the DBB and also for putting the pointers into the DRQ.

The DBB need not be contiguous. In fact, the DBB need not be in a fixed location. The local operating system does not have to process the incoming blocks as long as it can allocate enough free block and fill the DRQ with enough pointers.

The protocol is illustrated in Figure 2.2.

FIG 2.2 PROTOCOL BETWEEN GATEWAY CONTROLLER & THE LOCAL OPERATING SYSTEM



3.0 THE SIMULATOR

3.1. The Structure of the Simulator

The simulator consists of an event scheduler and a set of simulation procedures, each simulates the operation of a specific type of machine (processor, gateway controller, etc).

In the hardware environment of INFOPLEX, more than one machine (processor or gateway controller) can be active at the same time. This concurrency is simulated by dividing the operation of each machine into time slices, and by having the simulator executing the time slices of each machine in an interleaving manner.

3.1.1. Synchronous and Asynchronous Activity Handlers

Because in reality, a processor running a user program can be interrupted to handle an incoming message or serve other operating system functions, user programs and operating system functions can run 'simultaneously' within the same processor. Thus, for convenience and clarity, each processor is divided logically into two parts: an Asynchronous Activity Handler (AAH) and a Synchronous Activity Handler (SAH). Events which can happen and must be handled asynchronously with the scheduling cycle (of the operating system), such as the arrival of a message from another level, are handled by the AAH. (In essence, the AAH is like an interrupt

handler.) Scheduling and running user programs, and other events which could be conveniently processed at scheduling cycle breaks, are handled by the SAH. These two parts are treated by the simulator as if they were two separate and independent machines.

This separation bases on the assumption that the time consumed by handling asynchronous events is much smaller than that consumed by synchronous events, so that the work of the AAH would not slow down the SAH. The performance data gathered during simulation (see Chapter 7) will be used to verify this assumption.

3.1.2. Events

In many cases, during a time slice, a machine may stay idle indefinitely waiting for some events to happen. For example, a gateway controller idles until either an incoming or outgoing message arrives. It would be wasteful to simulate this kind of indefinite loop. Moreover, the simulator has to stop the simulation of the waiting machine at some point so that it can run the time slice of another machine to materialize the expected event. Therefore, time slices are scheduled only when a machine has something meaningful to do.

To formalize this concept, we say that time slices are triggered by events. An event is a condition which causes a machine to do something, and will usually cause other events to occur. The machine is known as the ser-

ver or the handler of the event.

3.1.3. Event and Server Queues

To keep track of all the events in a system, the simulator maintains a set of event queues. Events in a queue are ordered by the time they occured. An event can appear in only one queue and the grouping of events into queues is such that two events are put in the same queue if and only if they can be handled by the same set of machines (servers).

The simulator also maintains a set of server queues, which keep track of when each machine will be available to handle an event. (It is assumed that a machine can handle only one event at a time.) A machine can appear in only one server queue and the grouping of machines into queues is such that two machines are in the same queue if and only if they handle the same set (i.e., type) of events.

In the current configuration of the simulator, there are three event queue - server queue pairs for each level.

In the first server queue, the gateway controller is the sole server. The corresponding event queue contains the events which are the arrivals of messages to the gateway controller, from either the global bus or the processors of the local operating system at that level.

In the second queue pair, the server queue contains the AAH's of the level. The event queue contains the events which are the arrival of messages at the local operating system. (The messages are already in the DBB or the SRQ, being put there by the gateway controller.)

In the third queue pair, the server queue contains the SAH's of the level. The event queue contains several types of events: LOGMSG, SYNC, WAIT, FINISH and INIT. An event of type LOGMSG is the arrival of a logical message at the mail box of one of the virtual processes. A logical message is a logical unit of data transfer between two virtual processes, and can be queued by either the SAH or the AAH. Events of type SYNC, WAIT and FINISH are issued by virtual processes to control the scheduling process. Events of type INIT are queued by the simulator at the beginning of simulation for initializing the local operating system. (Chapter 5 will deal with the operation of a local operating system in detail.)

In addition to the queues mentioned above, the simulator also maintains an event queue for the arrival of messages at the global bus. The global bus simulator is the sole server in the corresponding server queue.

3.2. Operation of the Simulator

The simulator runs by simulation cycles. At the beginning of each cycle, the start time of the event at the head of each event queue is calculated.

The start time of an event is the earliest time at which a machine is available to handle the event. This may be later than when the event occurs because there may not be any free machines then. The simulator selects the earliest serviceable event and 'dispatches' the server machine by executing the procedure which simulates the machine. The event is removed from the queue and is passed to the simulation procedure as an argument.

The simulation procedure will most probably cause more events to occur. Simulation will stop when there are no events in all the queues.

For each machine, the simulator keeps track of how far into simulation time (STIME) scale the machine has run in an STIME clock. The simulation procedure for the machine is responsible for updating the STIME clock to reflect the operating speed of that machine.

When the simulation procedure returns, it will be requeued into the server queue with its updated STIME clock. The simulator will then goes into another cycle.

3.3. SEND

SEND is the procdure which simulates the action of a processor in transfering data to the gateway controller. Further description of SEND is found in Chapter 5.

When a virtual process wants to send a message to another level, it calls the procedure SEND. SEND queues the message as an event of type MSGOUT in the event queue of the local gateway controller and stamps it with the current STIME clock of the processor plus a delay parameter. The STIME clock of the processor is then incremented by an amount which is a function of the length of the message and an overall throughput parameter.

Note that this simulation algorithm corresponds to three different scenarios of how the message is sent in reality. In the first one, when the user program wants to send a message, the processor immediately gains access to the local gateway controller and performs the data transfer. In the second one, the message from the virtual process is queued by the operating system in the memory while it keeps trying to get access to the gateway controller. Within a reasonable delay, the operating system gains control of the gateway controller and transfer the message. In the third scenario, the message is queued in a specific place in the memory from where it will be picked up by the gateway controller at

a later time without the intervention of a processor. By varying the delay parameter, the effect of all of the three alternative configuartions can be satisfactorily simulated.

3.4. GCER: Outgoing Messages

GCER is the procedure which simulates the action of a gateway controller. For an outgoing message, GCER queues it in the event queue of the global bus and stamps it with its current STIME clock of the gateway controller plus a delay parameter. The STIME clock of the gateway controller is then incremented by an amount which is a function of the length of the message and an overall through put rate parameter.

Note that since GCER handles only one message (either outgoing or incoming) at a time, the start time of an event handled by GCER can be considerably later than the sending time of the message. It is assumed that this delay in the start time is small enough so that it would not affect the sending machine. To verify this assumption, GCER keeps track of the average delay in start time for both incoming and outgoing messages.

3.5. GBER

GBER is the simulation procedure for the global bus. GBER queues incoming messages to the event queues of their respective destination gateway controllers and

stamps them with their arrival time plus a delay parameter. Since the global bus can handle interleaved messages, the service delay is assumed to be always zero and the STIME clock of the global bus is set to the arrival time of each message.

It is assumed that the overall effect of the global bus between the sending and receving gateway controllers is a simple delay with a reasonable standard deviation. This assumption is valid if the time distribution of the messages is sparse enough and that the nature of the pended bus will even out the through put rate of messages at different load conditions of the global bus. To verify this assumption, GBER keeps track of the arrival time and duration of each message.

3.6. GCER: Incoming Messages

GCER handles an incoming message in a way similar to that of handling an outgoing one. It queues the message in the event queue of the local AAH's, and time stamps it with the current STIME clock plus an amount which is a function of the length of the message and an overall through put rate parameter. The STIME clock is incremented by the same amount.

Note that it is assumed that the gateway controller can handle interleaved incoming messages.

3.7. AAHER

AAHER is the procedure which simulates the action of an AAH. AAHER removes messages from the SRQ or the DBB and reassembles them if necessary to form logical messages. It checks to see if the logical message is addressed to the operating system itself: if it is, AAHER will process it immediately. Otherwise it will be queued in the event queue of the local SAH's with type equal to LOGMSG and STIME equal to the STIME clock of the AAH. The LOGMSG event will be handled by the SAHER.

3.8. SAHER

SAHER is the procedure which simulates the action of a SAH. A full description of its operation is given in Chapter 5.

In short, each activation of SAHER resembles a scheduling cycle (of the operating system): logical messages are distributed to their respective mailboxes, the waiting condition of each process is then evaluated, and if there is any runnable process, one is selected and executed.

3.9. Data Transfer

In the simulator, a message is represented by a POINTER to a data structure containing the content of the message. The simulator does not access the data structure directly, and a message transmission is

simulated by passing the pointer from one place to another.

Also, the simulator does not maintain the SRQ's, the DRQ's or the DBB's. Instead, for each level, the simulator keeps track of the size of a virtual SRQ and that of avirtual DBB. When a S-type message arrives to a local operating system, the size of the virtual SRQ will be incremented (by the length of the S-type message); and when a message is processed, the size will be decremented. Thus, the simulator knows how much data is supposed in the SRQ even though there is no copying of data involved. D-type messages are treated similarly, except that the size of a DBB is kept in terms of blocks (not bytes).

4.0 THE LOCAL OPERATING SYSTEM

4.1. Overview

The local operating system at each level is a multi-processor, multiprocess operating system. Each processor shares the duty of scheduling and handling inter-process communication. There is no one processor which is dedicated to running system programs. Operating system data such as the status of each virtual process reside in the shared memory and can be accessed by all the processors. Each processor schedules its own activities and updates the shared data.

Each virtual process in the local operating system has its own virtual addressing space and is identified by a unique ID number (VPID). A process communicates with another by sending and receiving logical messages. The receiving process can be in the same level (intra-level mail) or at another level (inter-level mail).

A logical message can be broken up into several smaller physical messages to be transmitted in intervals to lighten the load on the global bus or the gateway controller. In this case, the receiving operating system is responsible for reassembling the incoming messages to recover the original logical message).

At present, SHELL does not break up logical messages for transmission.

A process has an array of virtual mail boxes. Each mail box is identified by a mail box number and can hold more than one logical message. The address of a logical message identifies the level, the VPID and a mail box number of the destination process. When a process is expecting a message, it specifies to the operating system to which mail box is the expected message addressed. It is then put into a suspended state until a message arrives at that mail box. It will then be waken up by the operating system.

The format of a message and to which mail box it should be sent is settled entirely between the sender and receiver process. Usually, when a process expects a reply, it will send a mail box number as part of the message, indicating to which mail box the reply should be sent. By convention, box No. 1 is used by the operating system to pass data to a process when it is initially created.

4.2. System Service Requests

Virtual process 0 (zero) is recognized as the operating system itself. A process can request a service from an operating system by sending a message to virtual process 0. The request can be made to the operating system at the local level or that at a foreign level. In either case, the box number will be decoded as the type of service requested. Currently, only one

type of system service is recognized: creating a new process with a user specified top level procedure.

4.3. The Local Operation System as a Network of Co-Routines

The set of virtual processes in essence forms a network of concurrent coroutines (or modules, as in MODULA [Wirth]). There are two general approaches of allocating co-routines to perform the different functions required by INFOPLEX. One is to allow each transaction to have separate co-routine instances, thus forming a set of parallel, non-intervening chains of activations. The advantage of this arrangement is that the co-routines can be made to fit the characteristic of each individual transaction, and pipeling can be more readily realized.

The other approach is to arrange a co-routine instance to handle a data structure, possibly being shared by several transactions. The co-routine in this case will act as the data handler and the arbiter (as in the MONITORs of Hoare [Hoare]). The advantage of this arrangement is that the co-routines can be made to fit the characteristics of each data structure. In this case, starting a new process is analogous to opening a file, and subsequent requests to the co-routine is like accessing a file which is already opened. After the initial set up, records are retained by the co-routine

so that subsequent processing can be more efficient.

The structure of the operating system allows the writer of application programs to experiment with both approaches.

5.0 THE LOCAL OPERATING SYSTEM EMULATOR

5.1. Overview

The local operating system emulator performs the following functions: 1) scheduling and executing processes, and providing the basic context switching mechanism; 2) inter-process communication; 3) inter-process synchronization; and 4) handling system service requests.

5.2. Process Control

5.2.1. Context Switching

In the current implementation, a process is the execution of a PL/1 procedure. Different processes running the same code corresponds to different instances of the same PL/1 procedure with different arguments. When a process is suspended, everything on the stack -- the AUTOMATIC variables, subroutine call records, the exception handling blocks -- are copied by the local operating system emulator to a storage area, and the stack space is cleared for the activation of the next process. When the process is resumed, the content of the stack at the time of its suspension are restored. The saving and restoring of the stack is completely transparent to the process. (The context switching mechanism will be described in detail in Chapter 6.)

Because STATIC and CONTROLLED variables are not preserved between context switching, the use of these types should be exercised with great caution.

5.2.2. The Virtual Process Status Table

The operating system maintains a Virtual Process Status Table (VPST) to keep track of the execution of each process. Each process is given a status variable, which can take the following values: RUNNING, BLOCKED, RUNNABLE, NASCENT or VOID. RUNNING indicates that a process is currently executing under one of the processors at that level. BLOCKED indicates that the process is suspended and cannot be scheduled because it is still waiting for a message. RUNNABLE indicates that a process is not currently running but can be scheduled in the next cycle. NASCENT is a special form of RUNNABLE and is used when the process has never been run before. (The distinction between NASCENT and RUNNABLE is necessary because starting a new process is handled quite differently from resuming an existing process. See Chapter 6 for details.) VOID indicates that the process has finished and that its slot in the VPST can be recycled.

In addition to the status, the VPST also keeps track for each virtual process: 1) the saved content of the stack; 2) how much time each process has run (VTIME); 3) the waiting box number; and 4) the list of

arrived mail

5.2.3. Scheduling

SAHER is the procedure which schedules and executes the virtual processes. A scheduling cycle is initiated whenever SAHER is dispatched to handle an event. When SAHER finished handling that event, it checks the list of incoming mail for each process to see if there is any mail going to the waiting box. If the waiting box has mail, the status of the process will be reset to RUNNABLE.

After checking the mail, SAHER will select from the set of RUNNABLE or NASCENT processes one process to run. In the current implementation, the scheduling algorithm is to choose the one with the least VTIME.

5.2.4. Process Termination

When a process has finished its task and wants to terminate, it calls the procedure FINISH and then does a RETURN to the calling procedure. FINISH queues an event which is of type FINISH and has the current STIME. The purpose of this event is to synchronize modification to the VPST (see below for details). The status of the process will be reset to VOID and its slot will be recyled.

5.3. Inter-Process Communication

5.3.1. Outgoing Mail: SEND

A process sends a message by calling the procedure SEND. It should specify as arguments to SEND: the level, VPID and mail box number of the destination process. The message is passed as a pointer to a data structure (whose format is of concern only to receving process). SEND checks to see if the message is addressed to a foreign level. If it is, SEND will queue it as a MSGOUT event to the event queue of the gateway controller of the level. If the message is addressed to the local level, SEND further checks to see if it is a system service request. If it is, the process is first synchronized (see below for detailed explaination), after which the service request will be processed immediately. If the message is a local mail to another process, it will be queued as an LOGMSG event to the local SAH's.

5.3.2. Incoming Mail

An incoming message to a level is first handled by an AAH, which queues it as LOGMSGs to the event queue of the SAH's. If the message is an operating system request, it is handled immediately by the AAH.

SAHER receives the LOGMSG's and puts them in the respective mail boxes. An error is flagged when a

message is sent to non-existing virtual process.

5.3.3. Indefinite Wait

When a process is expecting a message, it calls the procedure WAIT and passes to it as argument the waiting box number. WAIT will queue a WAIT event with the current STIME. (The purpose of the WAIT event is to synchronize the modification to the VPST -- see below for details.) The process is then suspended within the call to WAIT. When the waiting box has mail, WAIT will return.

5.4. Inter-Process Synchronization

Because the local operating system emulator executes in only a simulated multi-processor environment, the actual execution order of a series of operations by different processes may differ from the order which would be carried out if the operating system were in the real world. This can produce results that are inconsistent with the real environment.

To avoid this problem, one must make sure that an operation which is to be executed at STIME = T will be carried out only after all other operations which are to be excuted before T have been executed. The operation is then termed synchronized.

Not all operations need synchronization, but any access to data structures shared among processes should

be synchronized. Note that inter-level communication is implicitly synchronized.

5.4.1. SYNC

When a process wants to synchronize an operation, it precedes the operation with a call to the procedure SYNC. SYNC queues an event which is of type SYNC and has the current STIME of the SAH. The SYNC event will be handled after all other events with earlier STIMEs have been handled, thus ensuring a strict STIME chronological order. In the mean time, the process is suspended but its status is still RUNNING. When the SYNC event arrives, SAHER changes the status to RUNNABLE and the process is eventually rescheduled, at which point SYNC will return to its caller.

It is important that any access to a data structure shared among processes (e.g. a lock or a semaphore) be synchronized. In the operating system emulator, all the modification to the VPST are synchronised. WAIT and FINISH are special cases of SYNC which change the status of the issuing process to BLOCKED and VOID, respectively.

5.5. System Initialization

At the beginning of simulation, an event type of INIT is placed by the simulator in the event queue to the SAH's of the lowest level. This is analogous to a

'power-on' interrupt to the processors. Its purpose is to trigger the procedure which is responsible for initializing the local operating system.

6.0 PROGRAM STRUCTURE AND EXECUTION LOGIC OF SHELL

6.1. Program Structure of SHELL

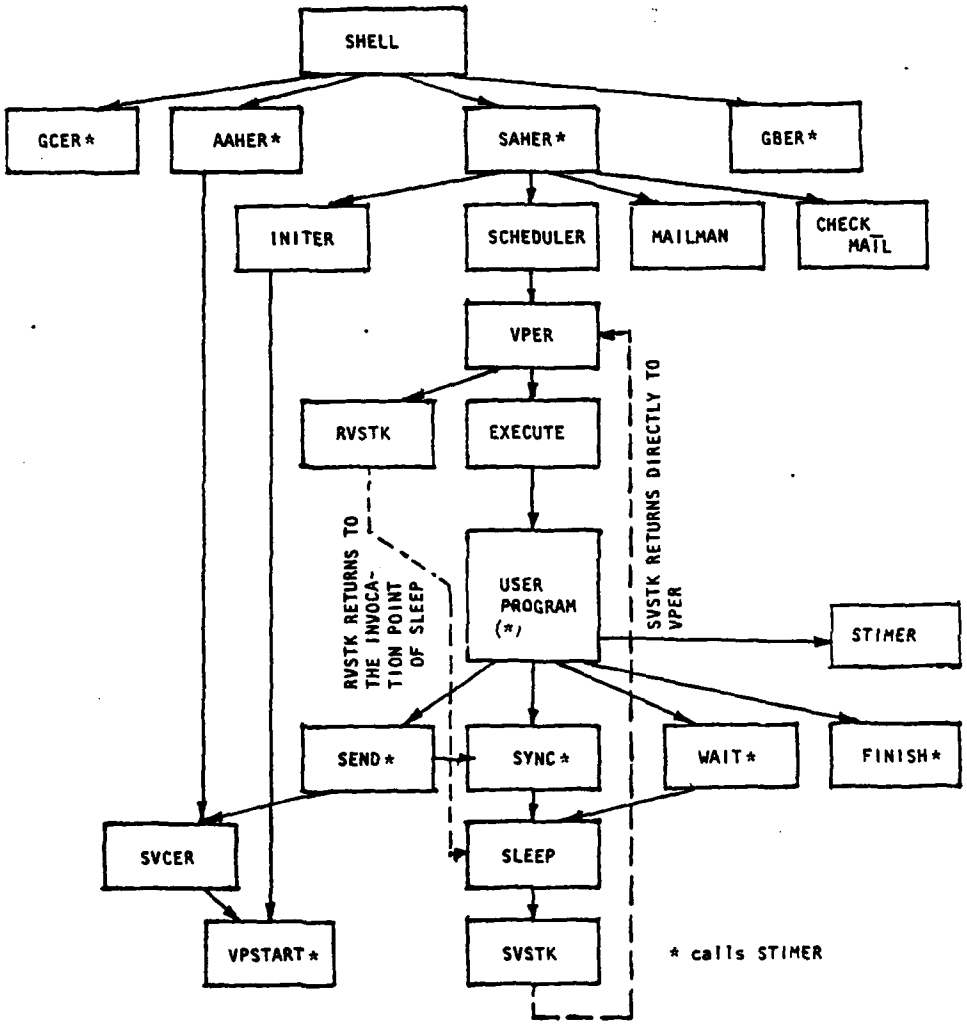
The functional relationship among the major modules of SHELL is depicted in Figure 6.1. SHELL, the event scheduler, dispatches events to one of the four simulation procedures GBER, GCER, AAHER and SAHER. Their functions are described in Chapter 3.

SAHER carries out the majority of the tasks of an operating system. (The other tasks are carried out by AAHER.) It in turn may call its own subroutines. Events of type SYNC, WAIT, and FINISH are handled by SAHER. For events of type LOGMSG, MAILMAN is called to distribute the mail. INITER is called to handle the event type INIT.

After incoming events are handled, SAHER calls CHECK_MAIL to check if any waiting virtual process has received its expected message. It then calls SCHEDULER to select a virtual process to run.

SCHEDULER in turn calls VPER to handle the virtual process. For a NASCENT process, VPER calls EXECUTE which calls the top level procedure of the process. For an old process, VPER calls RTSTK to restore the stack. RTSTK returns directly into the middle of the invocation of the newly restored process.

FIG. 6.1 FUNCTIONAL RELATIONSHIP OF MAJOR MODULES OF SHELL



The user program may call any of the utility procedures STIMER, SEND, WAIT, SYNC and FINISH. STIMER and FINISH both execute and return immediately. SEND may call SYNC if a system service request is involved, otherwise it should return immediately. SYNC and WAIT both call SLEEP, which in turn calls SVSTK to save the stack. SVSTK returns directly to VPER.

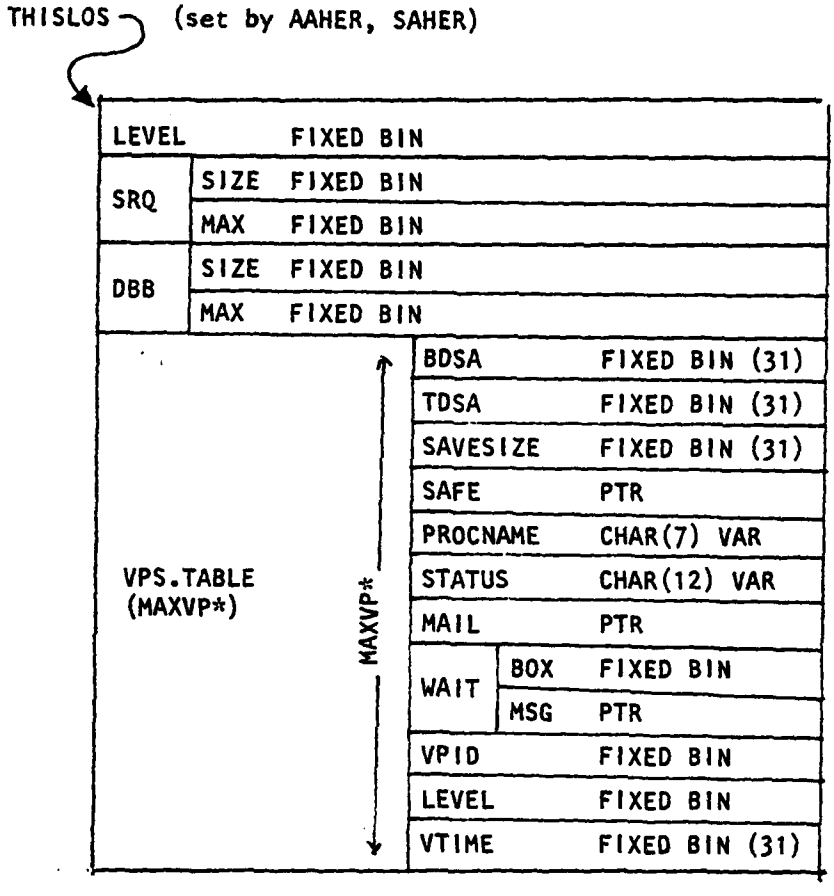
6.2. Format of Major Data Structures

6.2.1. LOS

SHELL maintains a giant table which contains the status of all the local operating systems. The format of an entry in the table (a LOS) is shown in Figure 6.2.

LOS.LEVEL is the level of the local operating system. LOS.SRQ.SIZE keeps the current amount of data (in bytes) in the SRQ. LOS.SRQ.MAX keeps the record of the maximum size the SRQ has reached, and is a good reference for finding the optimal size of the SRQ. Similarly, LOS.DBB.SIZE and LOS.DBB.MAX keep track of the current size and maximum size (in number of blocks) of the DBB.

LOS.VPS.TABLE is the virtual process status table of the local operating system. The number of slots in it, MAXVP, is a compile time parameter defined in the macro file CONFIG.



* a compile time parameter, defined in CONFIG

FIG 6.2 FORMAT OF LOS (Local Operating System)

The external POINTER variable THISLOS points to the current LOS. A user program can access the data of the current LOS via THISLOS.

6.2.2. VP

Each entry in the LOS.VPS.TABLE represents a virtual process, and its format is shown in Figure 6.3.

The fields VP.BDSA, VP.TDSA, VP.SAVESIZE and VP.SAFE are used by the context switching mechanism, and their used will be discussed in Section 6.4. VP.PROCNAME holds the name of the top level procedure. VP.STATUS holds the status of the virtual process, which must be one of the following: VOID, NASCENT, RUNNABLE, RUNNING, and BLOCKED. (See Chapter 5 for details.) VP.MAIL points to a chain of mail boxes, each containing a chain of messages (Figure 6.5.2).

VP.WAIT.BOX holds the number of the waiting box. VP.WAIT.MSG points to the expected message when it arrives. It also holds the initial message passed to it by the system when the process is newly created. (See Chapter 7 for details.)

VP.VPID contains the ID of virtual process. VP.LEVEL holds the level number. VP.VTIME holds the cumulative CPU time of the process (in micro-seconds).

The external POINTER variable THISVP points to the current VP entry. A user program can access the data of current virtual process via THISVP.

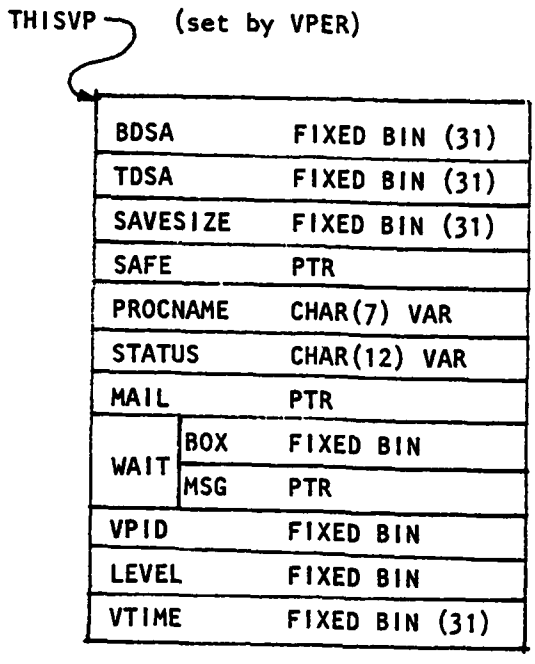


FIG 6. 3 FORMAT OF VP (Virtual Process)

6.2.3. SVR

The status of each machine (gateway controller, global bus, AAH, or SAH) in the system is kept in a SVR (Server). SVR's are chained together in the server queues kept by SHELL. (See Chapter 3.) The format of a SVR is shown in Figure 6.4.

SVR.NEXT points to the next server in the same server queue. SVR.STIME holds the base simulation time while SVR.RTIME holds the base real CPU time (in microseconds). SVR.STIMEQ holds the cumulative simulation time of the machine. SVR.RRATE is the ratio of the simulation clock speed to that of the real CPU clock (Chapter 7).

When a machine is dispatched to handle an event, SVR.STIME is set to the start time of the event, and SVR.RTIME is set to value of the real CPU clock at that time. Within the same time slice, the simulation time at any time t is given by:

(realtime(t) - SVR.RTIME) X SVR.RRATE + SVR.STIME

At the end of the time slice, SVR.STIME is updated to show when the machine will be available to handle the next event. Each type of machine has a different update algorithm.

The external POINTER variable THISSVR points to the current SVR. For a user program, this will always points to the SVR of a processor (SAHER).

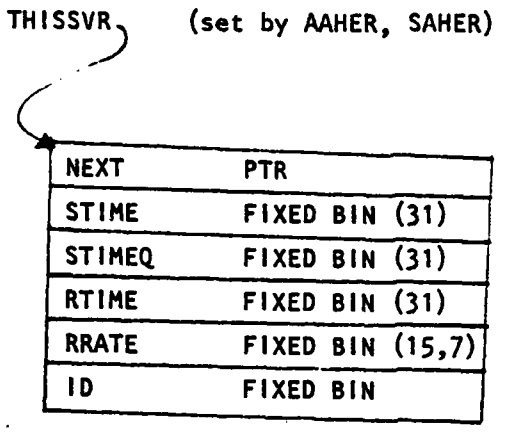


FIG. 6.4 FORMAT OF SVR (Server)

6.3. Message Prefixes

SHELL does not handle the data structure of a message, it only handles the POINTER to that data structure. At the beginning of a message transmission, several different types of data structures (prefixes) are chained to the beginning of the message. These prefixes are later on inspected and stripped off by the procedures in the receiving stages. Typically, a procedure looks only at the foremost prefix and does not care about the format of the data structures that follow.

The prefix convention reflects the nature of data transmission in the real world, where descriptor records are usually added to the beginning of a data stream to indicate its format or content.

Figures 6.5.1-2 depicts a pictorial history of a message transmission. The format of the prefixes involved are shown.

6.3.1. PF_ADDR

It contains the destination VPID and box number. It is added on by SEND and stripped off by MAILMAN.

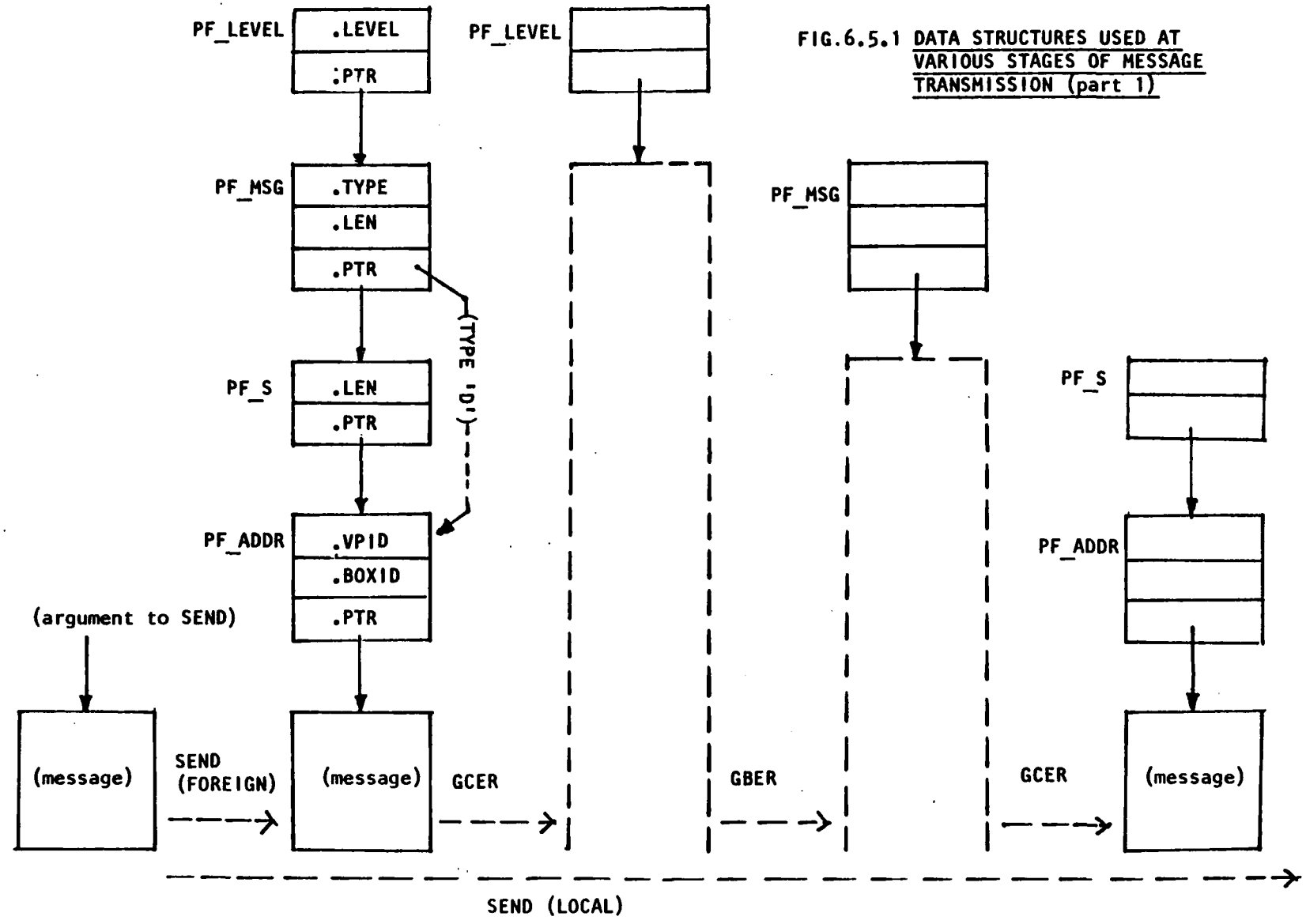


FIG.6.5.1 DATA STRUCTURES USED AT VARIOUS STAGES OF MESSAGE TRANSMISSION (part 1)

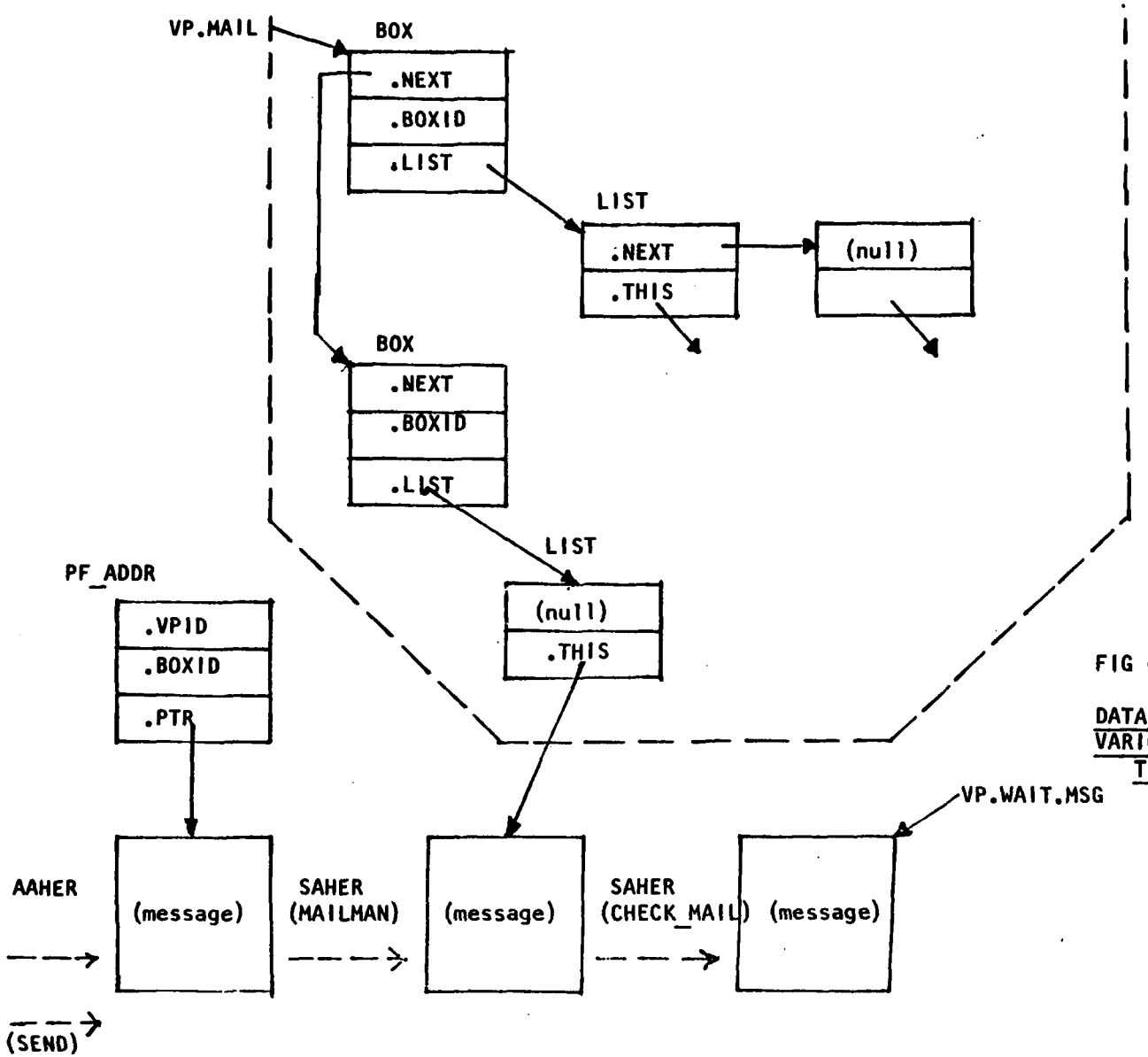


FIG 6.5.2

DATA STRUCTURES USED AT VARIOUS STAGES OF MESSAGE TRANSMISSION (part 2)

6.3.2. PF_S

It contains the length of the message (including that of PF_ADDR). Only S-type messages have it since D-type messages are fixed in length. It is added on by SEND. AAHER uses it to update the size of the SRQ, after which it is stripped off.

6.3.3. PF_MSG

It contains the type (either 'S' or 'D') of the message as well as its length (including that of PF_S and PF_ADDR).It is added on by SEND and stripped of by GCER.

6.3.4. PF_LEVEL

It contains the destination level of the message. It is added on by SEND. It is inspected by both GCER and GBER. It is stripped off by GBER.

6.4. The Execution Logic of the Context Switching Mechanism

The context switching mechanism works only in the IBM OS PL/1 execution environment. (For details of the environment, see [IBM].)

6.4.1. The Stack Space

Every time a PL/1 procedure is invoked, it allocates a temporary storage area on the stack. This area is headed by the DSA (Dynamic Storage Area) of the procedure, followed by the AUTOMATIC variables and other

temporary data. The DSA holds the NAB (Next Available Byte), a pointer to the end of the temporary storage area. If the procedure calls another procedure, the DSA will also hold the return address for the called procedure.

The stack space in the PL/1 execution environment is initially contiguous but may later become segmented. Each segment of the stack space is identified by a segment number which is also contained in the DSA. The context switching mechanism requires a contiguous stack space. Hence, every time before saving or restoring the stack, the segment number is checked to make sure that the stack space is still contiguous.

The end of the stack space is marked by the EOS (End of Segment) slot in the TCA (Task Communicaton Area) of the PL/1 execution environment. EOS changes as BASED or CONTROLLED storages are ALLOCATE'd and FREE'd.

The format of the DSA is shown in Figure 6.6, and the format of the TCA is shown in Figure 6.7. For further information about the internal data format of the PL/1 execution environment, see [IBM].

6.4.1.1. GET4

GET4 is an assembly language subroutine. It obtains the following information: the address of the DSA (of the calling PL/1 procedure), the NAB, the segment number and the EOS.

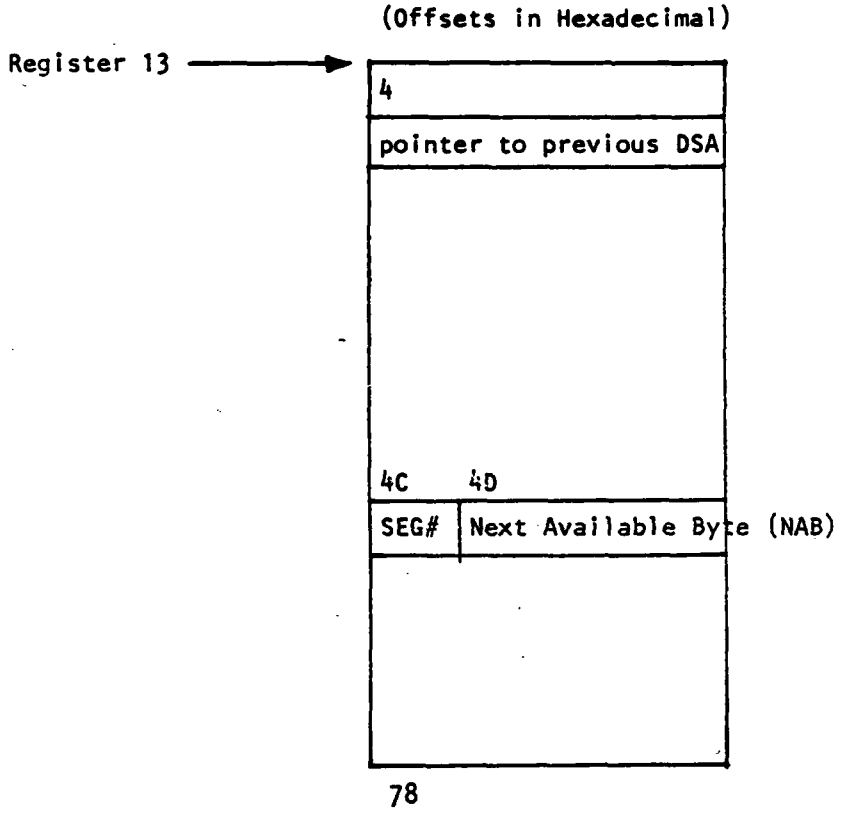


FIG. 6.6 FORMAT OF THE DYNAMIC STORAGE AREA (DSA)

(for further details of the format, see [IMB])

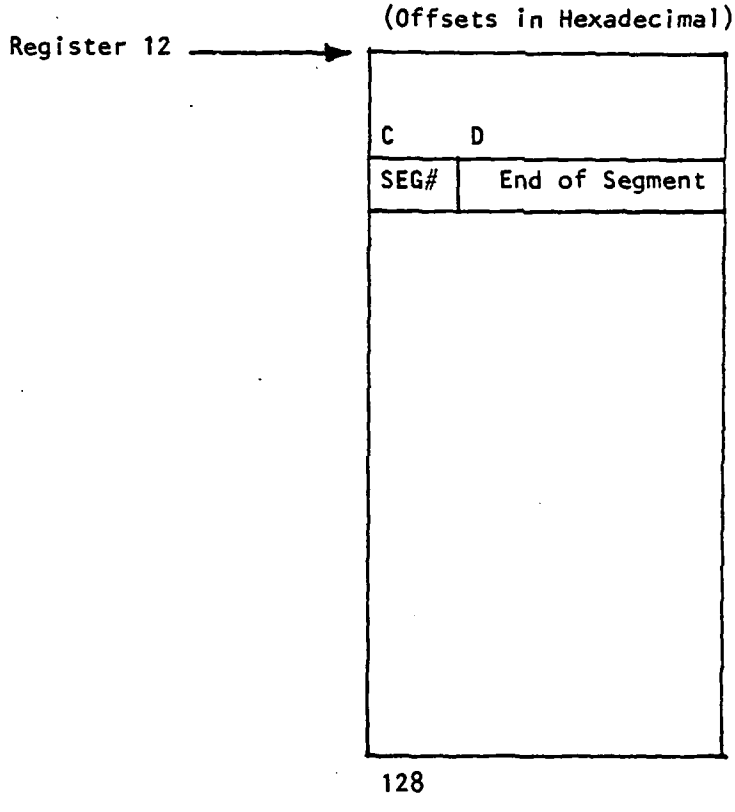


FIG. 6.7 FORMAT OF THE TASK COMMUNICATION AREA (TCA)

(for further details of the format, see [IMB])

6.4.2. Saving a Process

Before invoking a process, VPER first gets (via GET4) the address of its own DSA and puts it in the BDSA (Bottom DSA) field of the current VP. It then calls EXECUTE, which in turn calls the user program.

Control eventually reaches SLEEP. SLEEP gets (via GET4) the address of its own DSA and puts it in the TDSA (Top DSA) field of the current VP. The amount of data to be saved (between the BDSA and the NAB of the TDSA) is calculated and put in the SAVESIZE field of the VP.

SLEEP then allocates a number of save blocks, which are chunks of BASED storage chained together. VP.SAFE points to the first save block.

SLEEP then calls SVSTK. SVSTK copies the content of the stack (from the BDSA to the NAB of the TDSA) to the save blocks. SVSTK returns directly the the BDSA, as if EXECUTE were returning to VPER.

6.4.3. Restoring a Process

Before restoring a process, VPER checks (via GET4) to see if there is enough space left in the stack space for restoration. If not, an error will be signaled.

VPER then calls RTSTK, which copies the data from the save blocks to the stack space. Every datum is put back at the exact same address. Note that the DSA of VPER is 'painted over' by RTSTK, so that the state of

VPER is restored to the state when it called EXECUTE. RTSTK returns to the TDSA, as if SVSTK were returning to SLEEP. Control will eventually return to the user program. The saving and restoring is transparent to the user program; and as far as the user program is concerned, it called a utility procedure (SYNC or WAIT) and it has just returned.

The state of the stack space at various stages is depicted in Figures 6.8.1-4.

FIG. 6.8.1 STATE OF THE STACK DURING CONTEXT SWITCHING (part 1)

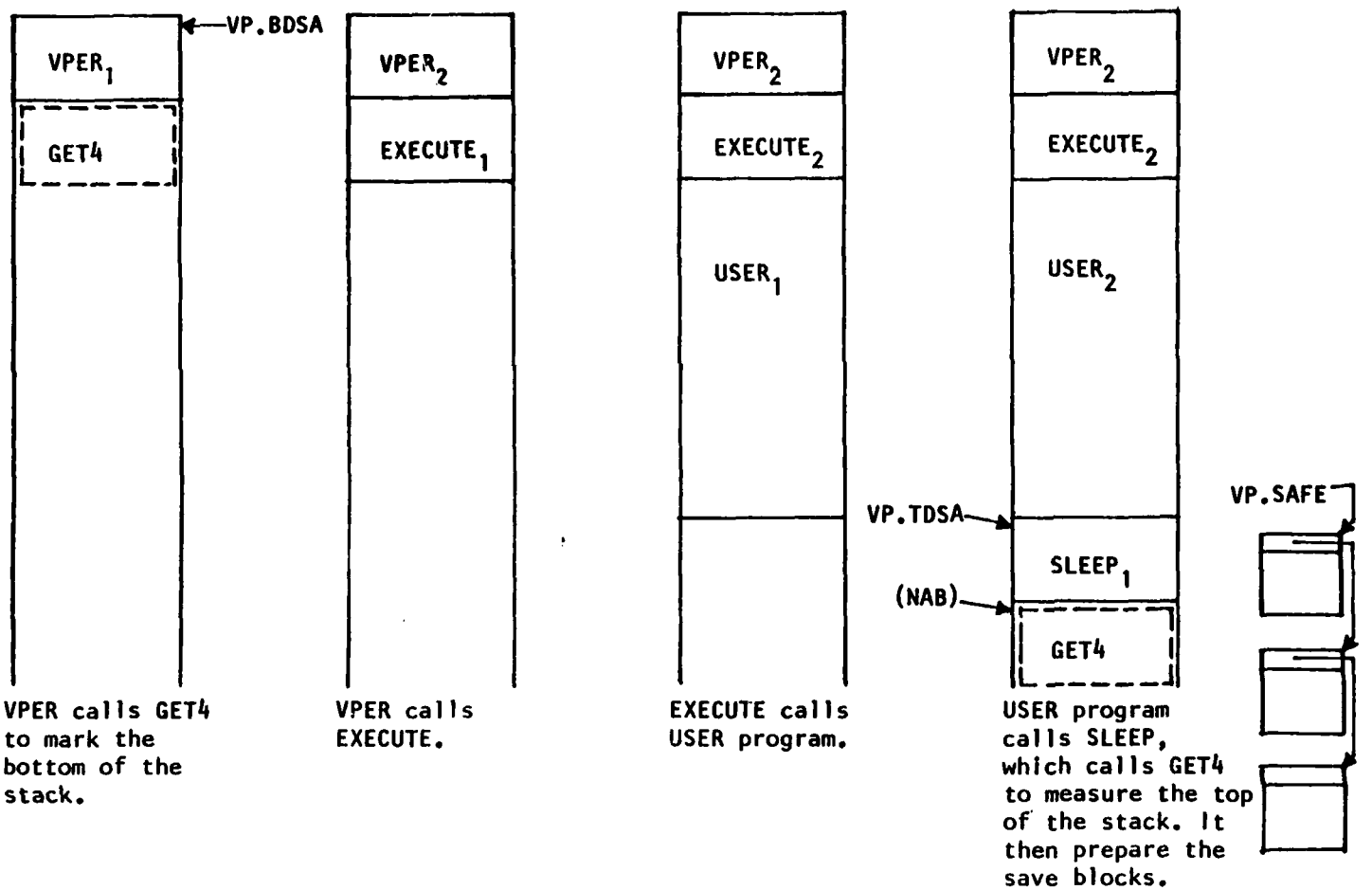


FIG. 6.8.2 STATE OF THE STACK DURING CONTEXT SWITCHING (part 2)

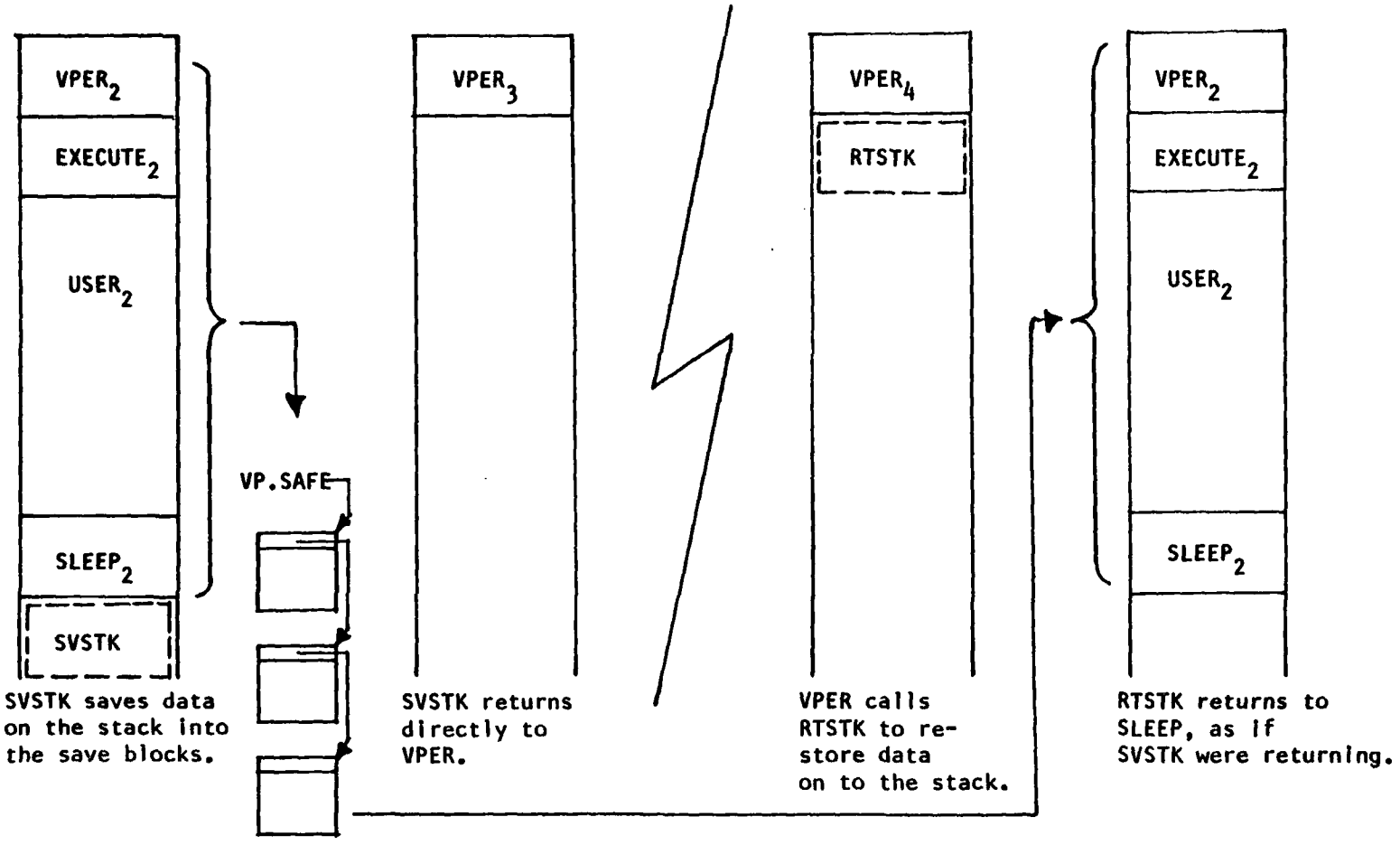


FIG. 6.8.3 STATE OF THE STACK DURING CONTEXT SWITCHING (part 3)

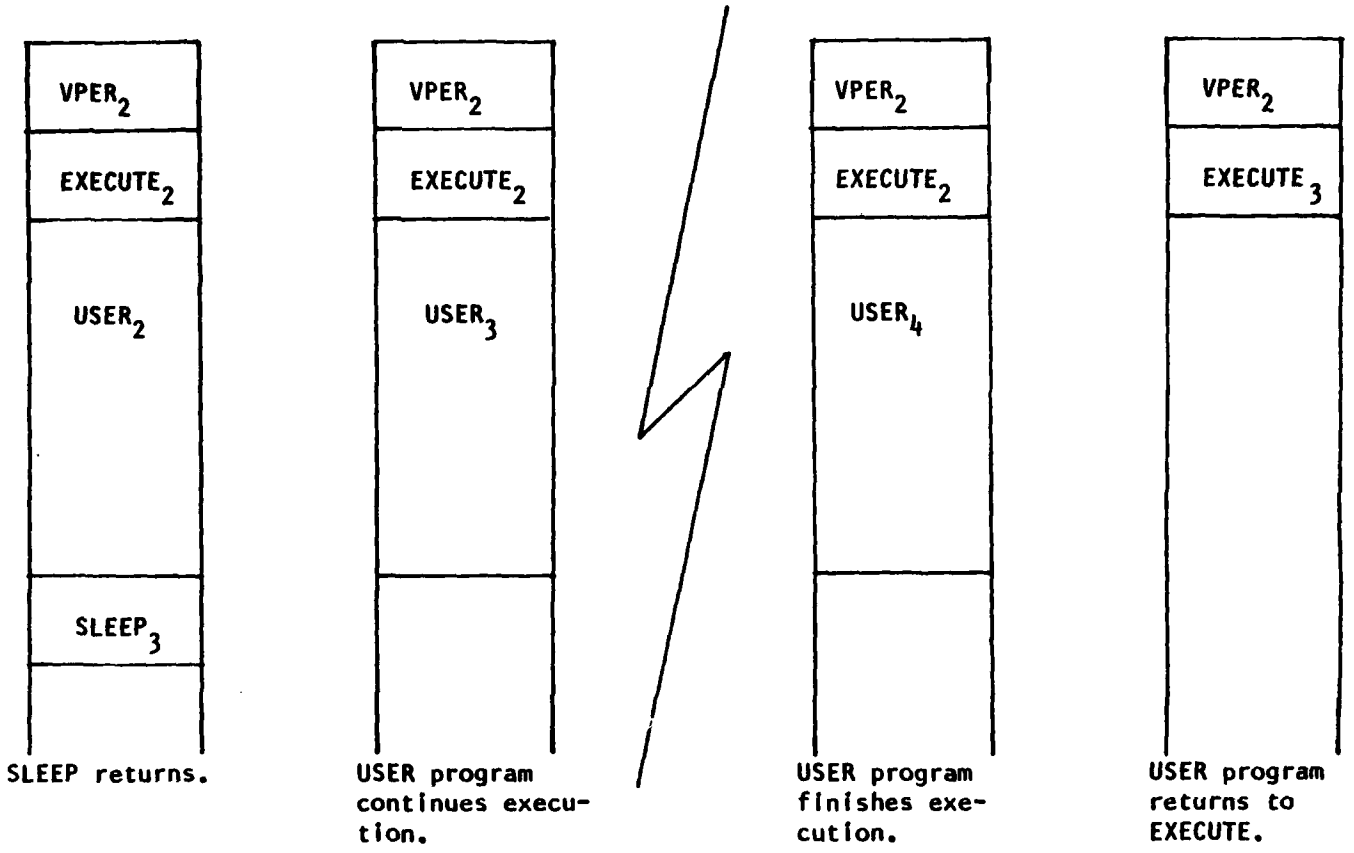
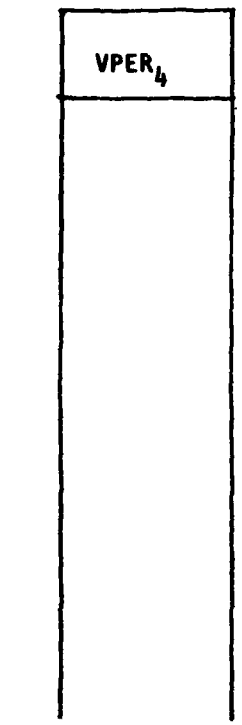
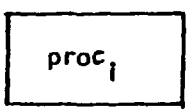


FIG. 6.8.4 STATE OF THE STACK DURING CONTEXT SWITCING (part 4)



EXECUTE returns.

Legend:



Activation Record of a procedure on the stack (DSA, AUTOMATIC variables, et al). The subscripts disguish the different states of the procedure.

Assembly language procedure which does not use the stack.

7.0 INTERFACE WITH APPLICATION PROGRAMS--A USER'S GUIDE TO SHELL

To adopt a PL/1 program to run under SHELL, it must first be re-organized to fit the multi-process (co-routine) execution environment of SHELL. In specific, it should be functionally decomposed into a set of top level procedures, each top level procedure being an independent unit performing a specific function. There should be no sharing of data or subroutines among top level procedures, but within each top level procedure, there can be internal subroutines and data can be shared among the internal subroutines.

Top level procedures communicate with one another only by sending and receiving messages, using the utility procedures SEND and WAIT. (Section 7.4)

A virtual process in SHELL is an activation of a top level procedure. Several processes may share the same top level procedure. A virtual process is created by SHELL upon request from other processes. A process can request a new process to be created at its own level or at another level. It does so by sending a message to the operation system of the desired level and specifying which top level procedure is to be used. (Section 7.5)

At the beginning of simulation, SHELL creates several processes with the user-supplied procedure TERM. (Section 7.3) TERM should be programmed to handle the initial terminal I/O and start the action sequence.

One starts the simulation by loading and running SHELL.

7.1. To Register a Top Level Procedure with SHELL

EXECUTE is the (only) procedure in SHELL that reference the user-supplied top level procedures. It is a very simple program which consists almost entirely of clauses of the form: (Figure 7.1)

```
IF (PROCNAM='X') THEN DO; CALL X; RETURN; END;
```

To add a top level procedure to SHELL, one should modify EXECUTE by adding a clause of the above form (with the appropiate name substituting the 'X') to the body of EXECUTE. EXECUTE should then be recompiled.

Table 7.1 shows a list of EXTERNAL variable and entry names used by SHELL. A user-supplied top level procedure should not have a name identical to the names in the list.

7.2. Parameters

7.2.1. Compile Time Parameters

There are only two compile time parameters: MAXLEVEL and MAXQ. MAXLEVEL is the one less the number of levels in the functional hierarchy; and MAXQ is always MAXLEVEL x 3. They are both defined in the macro file CONFIG. To set these two parameters, modify the macro file CONFIG and recompile SHELL.

SOURCE LISTING

```
NUMBER LEV NT

    10      0  EXECUTE: PROC (PROCNAME);                                    EXE00010
                                                                            EXE00020
    30   1  0  DCL PROCNAME CHAR(*) VAR;                                    EXE00030
    40   1  0  DCL TERM ENTRY;                                              EXE00040
                                                                            EXE00050
    60   1  0      IF PROCNAME = 'TERM' THEN DO; CALL TERM; RETURN; END;    EXE00060
                                                                            EXE00070
    80   1  0      PUT SKIP LIST (' ERROR: UNKNOWN PROCNAME. ',PROCNAME,    EXE00080
                                  ' (EXECUTE).');                           EXE00090
   100   1  0      STOP;                                                    EXE00100
                                                                            EXE00110
   120   1  0  END EXECUTE;                                                 EXE00120
```


FIG 7.1 LISTING OF PROCEDURE EXECUTE

Table 7.1 LIST OF EXTERNAL IDENTIFIERS

ENTRIES:	DATA:
AAHER	DEBUG
ADDQ	PARAMS
CONS	THISLOS
DEBUGR	THISSVR
EXECUTE	THISVP
FINISH	
GBER	PLIXOPT
GCER	
GET4	
QEVENT	
RTIMER	
RTSTK	
SAHER	
SEND	
SHELL	
SLEEP	
STIMER	
SVCER	
STSTK	
SYNC	
TERM	
VPER	
VPSTART	
WAIT	

7.2.2. Run Time Parameters

At the beginning of each simulation, SHELL asks for the values of a set of run time simulation parameters.

7.2.2.1. Number of Processors

Each level has a different number of processors.

7.2.2.2. RRATE

RRATE is the ratio between the simulation time clock rate and the real (CPU) time clock rate. It is declared as a FIXED BIN (15,7) variable. A RRATE greater than one means that the simulation time clock rate (of the processors) of that level is faster than the real CPU time rate (in 370 VM/CP), which means that the processor at that level are running slower than the real time clock. Each level has a different RRATE.

7.2.2.3. THRU_PUT

THRU_PUT is the overall average throughput rate at the global bus. The unit is in micro-seconds per byte. It is declared as a FIXED BIN (31,7).

7.2.2.4. DELAY_GB_GC, DELAY_GC_GB

DELAY_GB_GC is the average delay (micro-seconds) between the arrival of the first byte at the global bus and the arrival of the first byte at the destination gateway controller. DELAY_GC_GB is the counter part for outgoing messages.

7.2.2.5. TERMINALS

This is the number of 'virtual terminals' attached to the system. During initialization, SHELL starts that many number of virtual processes in the lowest level operating system. (See 7.3 for details.)

7.3. Initialization

At the beginning of simulation, the simulator puts into the lowest level operating system a number (= TERMINALS) of virtual processes with TERM as the top level procedures. TERM should be supplied by the user and should handle the initial user I/O and initialize the transactions.

7.4. Utility Procedures

There are five utility procedures which an application program can call directly: STIMER, SEND, SYNC, WAIT and FINISH. The macro file USERS contains the declarations of each of these entries and can be %INCLUDE'd to the user program.

7.4.1. STIMER

STIMER takes no argument and returns the current simulation time of current processor. It is controlled by the RRATE of the current level. The format of returned value is FIXED BIN (31).

7.4.2. SEND

The calling sequence of SEND is:

CALL SEND (level,vpid,boxid,type,len,ptr)

where level, vpid and boxid are those of the destination process. Type can be either 'D' or 'S', indicating which type of message should be sent. Len is the length, in bytes, of the message to be sent, and ptr is a POINTER pointing to the message. SHELL itself does not decode the message (except in the cases of system service requests, see Section 7.5); so the message can be of any data type as long as the receiving process knows how to decode it. Messages cannot be AUTOMATIC as the stack is modified when context switching occurs. (Ref Section 7.7)

7.4.3. SYNC

SYNC takes no argument. The calling process is suspended until it has become the earliest event in the system. (See Section 5.4 for details.) SYNC should be used before any access to a common shared data structure such as a lock or semaphore.

7.4.4. WAIT

WAIT takes a box number as argument. The calling process is suspended until a message arrives at the indicated mail box. If there is already a message in the mailbox when WAIT is called, it will have the same ef-

fect as SYNC.

7.4.5. FINISH

FINISH takes no argument and signals the end of the calling process. Note that FINISH does not transfer control out of the calling procedure, and a RETURN or a GOTO to the last END statement should follow the call to FINISH.

7.5. System Service Requests

Virtual Process 0 of each level is recognized as the operating system itself. A message sent to virtual process 0 will be decoded as a system service request. The destination box number in the case will be decoded as the type of system service desired.

Currently only one system service (box number = 1) is implemented. It is the starting of a new virtual process.

7.5.1. Start a New Virtual Process

The format of the message required by this system service is shown in Figure 7.2. A prefix (PF_SVC) is added to the beginning of the message which is going to be passed to the newly started virtual process. The prefix should contain the name of the top level procedure. After the operating system starts the process, the prefix is stripped off and the rest is put into the waiting box of the new process.

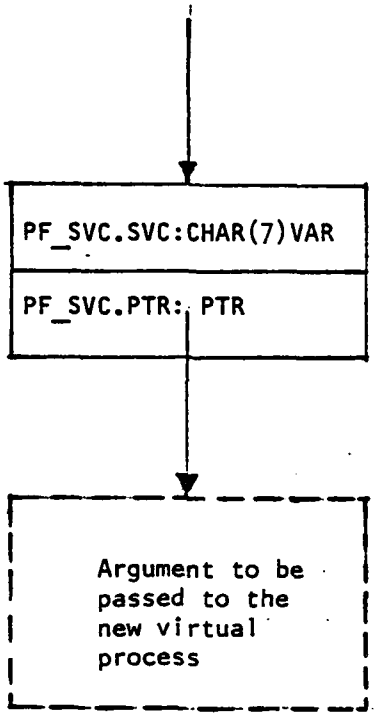


FIG.7.2 FORMAT OF ARGUMENT FOR SYSTEM SERVICE REQUEST 1 (STARTING A NEW VIRTUAL PROCESS)

7.6. External Variables

The EXTERNAL variables THISVP, THISLOS and THISSVR are POINTER varibles pointing respectively to the current Virtual Process, Local Operating System and Server (Processor). The format of data structures VP, LOS and SVR are listed in the Chapter 6. The macro files VPX, LOSX and SVRX contain all the necessary declarations and can be %INCLUDE'd by the application programs.

7.6.1. VP: Information about the current Virtual Process

Several sub-fields in VP are of special interest to an application program:

7.6.1.1. VP.WAIT.MSG

When a process wakes up *from* a WAIT, this slot contains the pointer to the message arrived at the expected mail box. When a process is newly created, this slot contains the argument passed to it by the system or the parent process.

7.6.1.2. VP.LEVEL

This is the level number of the embedded local operating system. It is declared as a FIXED BIN.

7.6.1.3. VP.VPID

This is the Virtual Process ID of the current process. It is declared as a FIXED BIN.

7.6.1.4. VP.VTIME

This contains the cumulative simulation time (micro-seconds) of each process, in the format of FIXED BIN (31).

7.7. Caveats & Restrictions

7.7.1. Local Variables

The AUTOMATIC variables in a top level procedure (and everything else that is stored on the stack) are associated with the process; while the STATIC and CONTROLLED variables are associated with the procedure. Hence, if a top level procedure is expected to be shared by more than one processes, it must either use only AUTOMATIC variables or keep track of its STATIC and CONTROLLED variables very carefully.

7.7.2. Messges

SHELL keeps track of messages as POINTERs and does not care about the actual data format or the storage class of the messages. However, because the stack space is multiplexed among processes, the AUTOMATC variables of a sending process will disappear from the stack space when the receiving process is invoked. Hence, messages

should not be contained inside AUTOMATIC data structures.

The prefered way is to use BASED data structures for messages. The sending process will ALLOCATE them and the receiving process will FREE them after processing them.

7.7.3. The REPORT option

Because the PL/1 run time option REPORT interferes with the operation of the context switching mechanism, it should never be used with SHELL.

7.8. STATS

STATS is a file containing certain performance statistics of SHELL. At present, it contains the following: 1) the run time parameters; 2) the average delay time of the gateway controllers; 3) the time distribution and duration of the messages arriving at the global bus; and 4) the cumulative active time of each machine.

7.9. A Sample Program

A test program is presented here as an example. It is called TERM so that it is activated by SHELL when simulation begins. A listing of TERM is in Figure 7.3 as well as in the Appendix.

When TERM comes up, it will identify itself by printing the current level, VPID, waiting box number, the ID of the processor (SVR), and the current STIME.

SOURCE LISTING

```
NUMBER LEV NT

    10       0  TERM: PROC;                                                          TER00010
                                                                                     TER00020
               %INCLUDE USERS;*************************************************TER00030
               %INCLUDE STIMER;************************************************USE00010
200010   1   0  DCL STIMER ENTRY RETURNS (FIXED BIN (31));                          STI00010
               ***************                                                       USE00010
300020   1   0  DCL SEND ENTRY (FIXED BIN, FIXED BIN, FIXED BIN, CHAR(*) VAR,      USE00020
                        FIXED BIN, PTR);                                           USE00030
300040   1   0  DCL WAIT ENTRY (FIXED BIN);                                        USE00040
300050   1   0  DCL SYNC ENTRY;                                                    USE00050
300060   1   0  DCL FINISH ENTRY;                                                  USE00060
                                                                                   USE00070
               ***************                                                     TER00030
400040   1   0  DCL (NULL,LENGTH) BUILTIN;                                        TER00040
                                                                                   TER00050
               %INCLUDE VPX;***************************************************TER00060
500010   1   0  DCL THISVP PTR EXTERNAL STATIC;        /* -> CURRENT VP, SET BY VPER */VPX00010
                                                                                   VPX00020
500030   1   0  DCL 1 VP BASED (THISVP), %INCLUDE VP;*****************************VPX00030
                     80 BDSA FIXED BIN (31),       /* ADDR (BOTTOM DSA)          */VP 00010
                     80 TDSA FIXED BIN (31),       /* ADDR (TOP DSA)             */VP 00020
                     80 SAVESIZE FIXED BIN (31),   /* SIZE OF AREA TO BE SAVED   */VP 00030
                     80 SAFE PTR,                  /* -> 1ST SAVBLK              */VP 00040
                     80 PROCNAME CHAR (7) VAR,     /* NAME OF TOP LEVEL PROC     */VP 00050
                     80 STATUS CHAR (12) VAR,                                      VP 00060
                     80 MAIL PTR,                  /* CHAIN OF INCOMING MAIL     */VP 00070
                     80 WAIT,                                                      VP 00080
                        81 BOX FIXED BIN,          /* BOX AWAITING MAIL          */VP 00090
                        81 MSG PTR,                /* MSG IN WAIT.BOX            */VP 00100
                     80 VPID FIXED BIN,            /* INDEX IN THE VPST          */VP 00110
                     80 LEVEL FIXED BIN,           /* LEVEL                      */VP 00120
                     80 VTIME FIXED BIN (31)                                       VP 00130
               ************************************    ;                          VPX00030
                                                                                   VPX00040
               *************                                                       TER00060
               %INCLUDE SVRX;**************************************************TER00070
800010   1   0  DCL THISSVR PTR EXTERNAL STATIC;   /* -> CURRENT SVR, SET BY SAHER */ SVR00010
                                                                                   SVR00020
800030   1   0  DCL 1 SVR BASED (THISSVR), %INCLUDE SVR;*************************SVR00030
                     70 NEXT PTR,                                                  SVR00010
                     70 STIME FIXED BIN (31),                                      SVR00020
                     70 STIMEQ FIXED BIN (31),     /* ACCUMULATED STIME          */SVR00030
                     70 RTIME FIXED BIN (31),                                      SVR00040
                     70 RRATE FIXED BIN (15,7),    /* REAL CPU TICK RATE         */SVR00050
                     70 ID FIXED BIN                                               SVR00060
               ************************************    ;                          SVR00030
```

FIG. 7.3.1 LISTING OF PROCEDURE TERM (part 1)

```
 NUMBER LEV NT

                                                                                SVR00040
                 ***************                                                TER00070
                 %INCLUDE PFSVC;*****************************************************TER00080
1100010   1  0   DCL PT_SVC PTR;                                                 PFS00010
                                                                                PFS00020
1100030   1  0   DCL 1 PF_SVC BASED (PT_SVC),                                    PFS00030
                       2 SVC CHAR (7) VAR,                                      PFS00040
                       2 PTR PTR;                                               PFS00050
                                                                                PFS00060
                 ***************                                                TER00080
                 %INCLUDE MSG;******************************************************TER00090
1200010   1  0   DCL MSGLEN FIXED BIN;                                          MSG00010
1200020   1  0   DCL MSGPT PTR;                                                 MSG00020
1200030   1  0   DCL 1 MSG BASED (MSGPT),                                       MSG00030
                       2 LEN FIXED BIN,                                         MSG00040
                       2 STR CHAR (MSGLEN REFER (LEN));                         MSG00050
                                                                                MSG00060
                 *************                                                  TER00090
1200100   1  0   DCL (LEVEL,VPID,BOXID) FIXED BIN;                             TER00100
1200110   1  0   DCL MESSAGE CHAR(80) VAR;                                      TER00110
1200120   1  0   DCL COMMAND CHAR(12);                                          TER00120
1200130   1  0   DCL TYPE CHAR(8) VAR;                                          TER00130
                                                                                TER00140
1200150   1  0   LOOP:                                                         TER00150
                     PUT SKIP EDIT (' TERM: LEVEL = ',VP.LEVEL,' VP = ',VP.VPID,  TER00160
                         ' WAIT.BOX = ',VP.WAIT.BOX,' SVR = ',SVR.ID,' STIME = ',  TER00170
                         SVR.STIME) (A,F(2),A,F(2),A,F(2),A,F(2),A,F(11));      TER00180
1200190   1  0       MSGPT = VP.WAIT.MSG;                                       TER00190
1200200   1  0       IF (MSGPT ^= NULL) THEN DO;                                TER00200
1200210   1  1          PUT SKIP EDIT (' MSG: ',MSG.STR) (A,A(MSG.LEN));        TER00210
1200220   1  1          FREE MSG;                                               TER00220
1200230   1  1          VP.WAIT.MSG = NULL;                                     TER00230
1200240   1  1       END;                                                       TER00240
                                                                                TER00250
1200260   1  0   WORK:                                                         TER00260
                     PUT SKIP;                                                  TER00270
1200280   1  0       DISPLAY (' COMMAND?') REPLY (COMMAND);                     TER00280
                                                                                TER00290
1200300   1  0       IF (COMMAND = 'BUILD') THEN DO;                            TER00300
1200310   1  1          PUT SKIP LIST ('  LEVEL> ');                           TER00310
1200320   1  1          GET LIST (LEVEL);                                       TER00320
1200330   1  1          MSGLEN = 7;                                             TER00330
1200340   1  1          ALLOCATE MSG;                                           TER00340
1200350   1  1          MSG.STR = 'NEW VPI';                                    TER00350
1200360   1  1          ALLOCATE PF_SVC;                                        TER00360
1200370   1  1          PF_SVC.SVC = 'TERM';                                    TER00370
1200380   1  1          PF_SVC.PTR = MSGPT;                                     TER00380
1200390   1  1          CALL SEND (LEVEL,0,1,'S',15,PT_SVC);                    TER00390
1200400   1  1          GOTO WORK;                                              TER00400
```

FIG. 7.3.2 LISTING OF PROCEDURE TERM (part 2)

```
 NUMBER LEV NT

1200410   1  1      END;                                                          TER00410
                                                                                 TER00420
1200430   1  0      ELSE IF (COMMAND = 'SEND') THEN DO;                           TER00430
1200440   1  1         PUT SKIP LIST ('  LEVEL,VPID,BOXID,TYPE,MESSAGE> ');      TER00440
1200450   1  1         GET LIST (LEVEL,VPID,BOXID,TYPE,MESSAGE);                  TER00450
1200460   1  1         MSGLEN = LENGTH (MESSAGE);                                 TER00460
1200470   1  1         ALLOCATE MSG;                                              TER00470
1200480   1  1         MSG.STR = MESSAGE;                                         TER00480
1200490   1  1         CALL SEND (LEVEL,VPID,BOXID,TYPE,MSGLEN,MSGPT);            TER00490
1200500   1  1         GOTO WORK;                                                 TER00500
1200510   1  1      END;                                                          TER00510
                                                                                 TER00520
1200530   1  0      ELSE IF (COMMAND = 'WAIT') THEN DO;                           TER00530
1200540   1  1         PUT SKIP LIST ('  BOX> ');                                TER00540
1200550   1  1         GET LIST (BOXID);                                          TER00550
1200560   1  1         PUT SKIP LIST ('  WAITING');                               TER00560
1200570   1  1         CALL WAIT (BOXID);                                         TER00570
1200580   1  1         GOTO LOOP;                                                 TER00580
1200590   1  1      END;                                                          TER00590
                                                                                 TER00600
1200610   1  0      ELSE IF (COMMAND = 'SYNC') THEN DO;                           TER00610
1200620   1  1         PUT SKIP LIST ('  SYNCING');                              TER00620
1200630   1  1         CALL SYNC;                                                 TER00630
1200640   1  1         GOTO LOOP;                                                 TER00640
1200650   1  1      END;                                                          TER00650
                                                                                 TER00660
1200670   1  0      ELSE IF (COMMAND = 'FINISH') THEN DO;                         TER00670
1200680   1  1         PUT SKIP LIST ('  FINISHING');                            TER00680
1200690   1  1         CALL FINISH;                                               TER00690
1200700   1  1         RETURN;                                                    TER00700
1200710   1  1      END;                                                          TER00710
                                                                                 TER00720
1200730   1  0      ELSE DO; /* UNKNOWN COMMAND */                               TER00730
1200740   1  1         PUT LIST ('  ??');                                        TER00740
1200750   1  1         GOTO WORK;                                                 TER00750
1200760   1  1      END;                                                          TER00760
                                                                                 TER00770
1200780   1  0   END TERM;                                                       TER00780
```


FIG. 7.3.3 LISTING OF PROCEDURE TERM (part 3)

If there is a message in the waiting box, it will print the message as well. TERM then asks for commands from the user and executes them. It recognizes five commands: BUILD, SEND, WAIT, SYNC and FINISH. After the command is performed, (it might take quite sometime), it will identify itself and ask for more work.

7.9.1. Command BUILD

BUILD creates a new process in the system with TERM as the top level procedure. It takes an argument, the level on which the new process is to be created. Note that at the beginning of simulation, there are already N (N = TERMINALS, a run time parameter) instances of TERM at level 0, being put there by the system initialization procedure. Repeated uses of BUILD can really proliferate the number of instances of TERM.

7.9.2. Command SEND

Command SEND takes five arguments: the destination level, VPID, and box number, the type of message (either 'S' or 'D') and the message itself (which must be within quotes). SEND will simply call (the utility procedure) SEND to send the message.

7.9.3. Command WAIT

Command WAIT takes one argument, the waiting box number. It will prompt 'WAITING..', and calls (the utility procedure) WAIT.

7.9.4. Commands SYNC and FINISH

Both these commands take no argument and are executed promptly by calling the corresponding utility procedure.

The network of instances of TERM is like a network of terminals or telex machines. The user can tailor the architecture of this network by building the desired number of processes at each level. Since the user can specify the action of each node in the network, any arbitary traffic pattern can be generated and tested.

7.10. A Sample Simulation Session

A very simple simulation session is scripted in Figures 7.4.1-3. The configuration in this simulation consists of two levels, each with two processors.

Simulation starts with three processes at level 0: VP0.1, VP0.2, and VP0.3. VP0.1 builds a process at level 1 (VP1.1), and then goes into waiting for a message to come to box 1. Similarly, VP0.2 builds VP1.2 then waits for box 2; and VP0.3 builds VP1.3 and waits for box 3.

```
R; T=0.01/0.01 01:08:47
shell
LOAD SHELL ( CLEAR NODUP NOMAP )
FILEDEF STATS DISK STATS LISTING A1 ( BLOCK 800 )
R; T=0.37/0.65 01:08:59
start
EXECUTION BEGINS...

 LEVEL  0:
  NO. OF PROCESSORS, RRATE>
:
2,1

 LEVEL  1:
  NO. OF PROCESSORS, RRATE>
:
2,1

 THRU_RATE,GB_GC,GC_GB,TERMINALS>

:
1,0,0,3


 START SIMULATION...



 TERM: LEVEL =  0 VP =  1 WAIT.BOX =  0 SVR =  2 STIME =          555
 MSG: SET UP BY INITER

 COMMAND?
build
  LEVEL>
:
1

 COMMAND?
wait
  BOX>
:
1

  WAITING


 TERM: LEVEL =  0 VP =  2 WAIT.BOX =  0 SVR =  1 STIME =         1062
 MSG: SET UP BY INITER

 COMMAND?
build
  LEVEL>
:
1

 COMMAND?
wait
  BOX>
:
```

FIG. 7.4 .1 SAMPLE SIMULATION SESSION (part 1)

```
2

  WAITING

 TERM: LEVEL =  1 VP =  1 WAIT.BOX =  0 SVR =  2 STIME =       58299
 MSG: NEW VP!

 COMMAND?
wait
  BOX>
:
1

  WAITING

 TERM: LEVEL =  1 VP =  2 WAIT.BOX =  0 SVR =  1 STIME =       70248
 MSG: NEW VP!

 COMMAND?
wait
  BOX>
:
2

  WAITING

 TERM: LEVEL =  0 VP =  3 WAIT.BOX =  0 SVR =  1 STIME =       95062
 MSG: SET UP BY INITER

 COMMAND?
build
  LEVEL>
:
1

 COMMAND?
wait
  BOX>
:
3

  WAITING

 TERM: LEVEL =  1 VP =  3 WAIT.BOX =  0 SVR =  2 STIME =      151919
 MSG: NEW VP!

 COMMAND?
send
  LEVEL,VPID,BOXID,TYPE,MESSAGE>
:
0,1,1,'S','from 1.3'

 COMMAND?
finish

  FINISHING

 TERM: LEVEL =  0 VP =  1 WAIT.BOX =  1 SVR =  2 STIME =      209840
 MSG: from 1.3

 COMMAND?
```

FIG. 7.4.2 SAMPLE SIMULATION SESSION (part 2)

```
send
  LEVEL,VPID,BOXID,TYPE,MESSAGE>
:
1,1,1,'D','from 0.1'

 COMMAND?
finisn

  FINISHING


 TERM: LEVEL =  1 VP =  1 WAIT.BOX =  1 SVR =  1 STIME =       267747
 MSG: from 0.1

 COMMAND?
send
  LEVEL,VPID,BOXID,TYPE,MESSAGE>
:
1,2,2,'S','from 1.1'

 COMMAND?
finisn

  FINISHING


 TERM: LEVEL =  1 VP =  2 WAIT.BOX =  2 SVR =  2 STIME =       324723
 MSG: from 1.1

 COMMAND?
send
  LEVEL,VPID,BOXID,TYPE,MESSAGE>
:
0,2,2,'D','from 1.2'

 COMMAND?
finisn

  FINI ING


 TERM: LEVEL =  0 VP =  2 WAIT.BOX =  2 SVR =  1 STIME =       384779
 MSG: from 1.2

 COMMAND?
send
  LEVEL,VPID,BOXID,TYPE,MESSAGE>
:
0,3,3,'S','from 0.2'

 COMMAND?
finisn

  FINISHING


 TERM: LEVEL =  0 VP =  3 WAIT.BOX =  3 SVR =  2 STIME =       441675
 MSG: from 0.2

 COMMAND?
finisn

  FINISHING


 R: T=0.64/1.76 01:11:57
```

FIG. 7.4.3 SAMPLE SIMULATION SESSION (part 3)

At level 1, VP1.1 waits for box 1 and VP1.2 waits for box 2. Then VP1.3 sends a message to VP0.1. After sending the message, VP1.3 terminates itself.

When VP0.1 wakes up, it sends a message to VP1.1 and then terminates. Similarly, VP1.1 sends to VP1.2; VP1.2 sends to VP0.2; and VP0.2 sends to VP0.3. When VP0.3 finally wakes up, it is the last process left in the system, and when it terminates, the simulation session ends as well.

In Figures 7.4.1-3, the user input is in lower case while the output of the simulator is in upper case. Figure 7.5 diagrams the chronology of the events in the simulation. The STATS file of the simulation is listed in Figure 7.6.

```
*** 81/05/19 01:09
*** PARAMETERS:

LEVEL  0: NP=        2 RRATE=    1.000;
LEVEL  1: NP=        2 RRATE=    1.000;
PARAMS.THRU_RATE=        1.000 PARAMS.DELAY_GB_GC=          0 PARAMS.DELAY_GC_GB=          0 PARAMS.TERMINALS=        3;

*** SIMULATION:

GC ( 0): MSGOUT DELAY =          0
GB: STIME =       57802 MSGLEN =         17
GC ( 1): MSGIN  DELAY =          0
GC ( 0): MSGOUT DELAY =          0
GB: STIME =       69750 MSGLEN =         17
GC ( 1): MSGIN  DELAY =          0
GC ( 0): MSGOUT DELAY =          0
GB: STIME =      151372 MSGLEN =         17
GC ( 1): MSGIN  DELAY =          0
GC ( 1): MSGOUT DELAY =          0
GB: STIME =      209459 MSGLEN =         10
GC ( 0): MSGIN  DELAY =          0
GC ( 0): MSGOUT DELAY =          0
GB: STIME =      267389 MSGLEN =         10
GC ( 1): MSGIN  DELAY =          0
GC ( 1): MSGOUT DELAY =          0
GB: STIME =      384414 MSGLEN =         10
GC ( 0): MSGIN  DELAY =          0

*** STATISTICS:

LEVEL  0:
  SRQ.MAX =         10 DBB.MAX =          1
  GC:
 SVR.ID=        1 SVR.STIME=       384424 SVR.STIMEQ=           81;
 AAM:
 SVR.ID=        2 SVR.STIME=       210193 SVR.STIMEQ=          724;
 SVR.ID=        1 SVR.STIME=       385128 SVR.STIMEQ=          704;
 SAM:
 SVR.ID=        1 SVR.STIME=       481150 SVR.STIMEQ=       266197;
 SVR.ID=        2 SVR.STIME=       481592 SVR.STIMEQ=       220819;
LEVEL  1:
  SRQ.MAX =         21 DBB.MAX =          1
  GC:
 SVR.ID=        1 SVR.STIME=       384424 SVR.STIMEQ=           81;
 AAM:
 SVR.ID=        2 SVR.STIME=       152346 SVR.STIMEQ=         1843;
 SVR.ID=        1 SVR.STIME=       268088 SVR.STIMEQ=         1582;
 SAM:
 SVR.ID=        1 SVR.STIME=       400088 SVR.STIMEQ=       134592;
 SVR.ID=        2 SVR.STIME=       400541 SVR.STIMEQ=       212533;
```

FIG. 7.5 STATISTICS OF SAMPLE SIMULATION SESSION

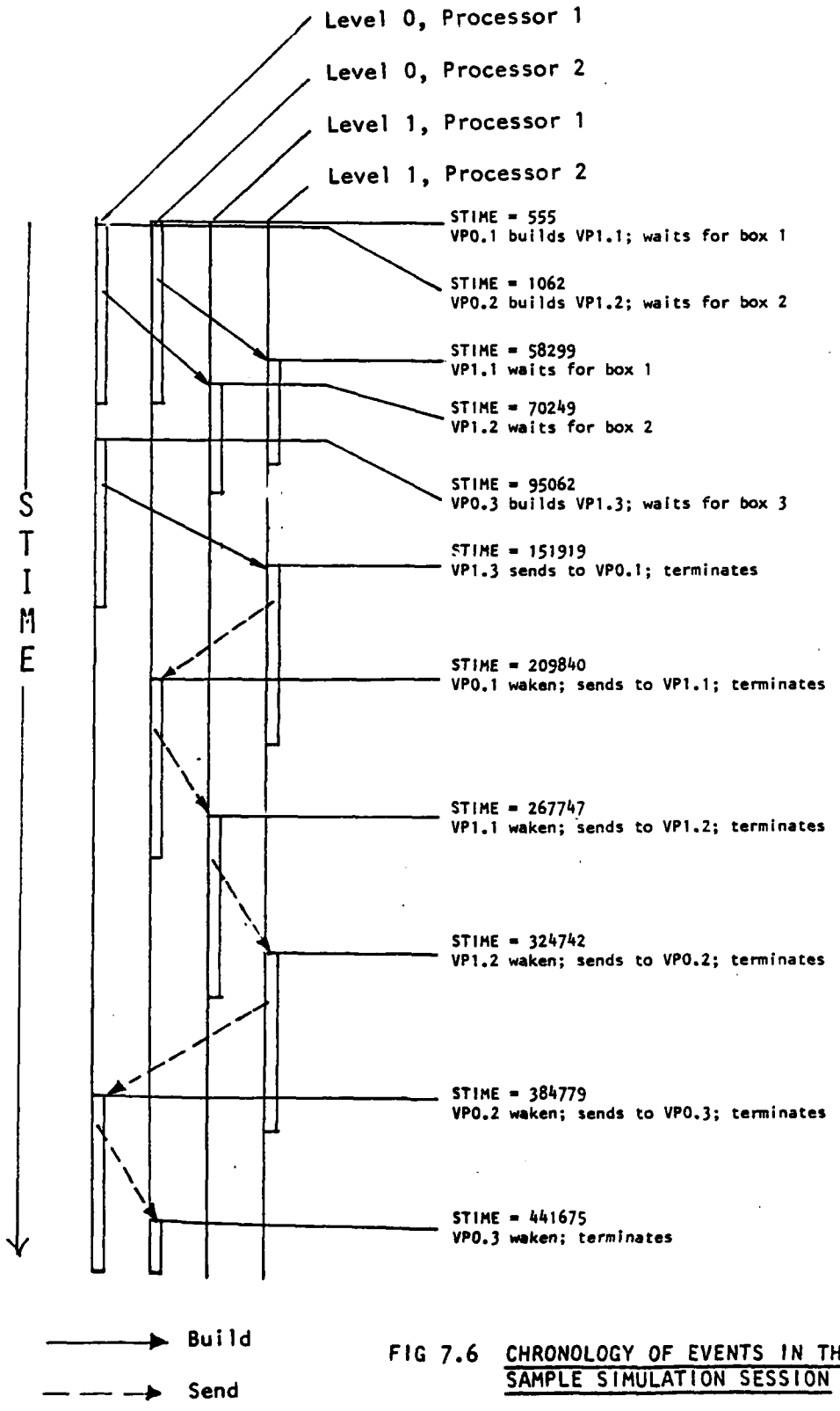


FIG 7.6 CHRONOLOGY OF EVENTS IN THE SAMPLE SIMULATION SESSION

8.0 CONCLUSION & OBSERVATIONS

SHELL is part of the first step in the gradual refinement of the design of INFOPLEX. As a simulator, it has two salient features.

The first is that SHELL offers a very easy interface with application programs. An ordinary PL/1 program can run under SHELL with minimal or no modification. The complete transparent context switching mechanism greatly eases the writting of application programs. Without this feature, an application would have to keep track of under which process it is working. Also, an application would not be able to exit (and return control to the simulator) from inside an inner block, since PL/1 does not allow control be transfered back into the inner block. This would severely handicap the programming style of the application program.

The second salient feature of SHELL is that it simulates a mixture of microscopic and macroscopic events. On the one hand, there are the short and simple events of data transaction among the machines; and on the other, the long and complex events of running application programs. This integrated approach offers a highly realistic simulation because one can study the interaction between the two types of events. For example, when one is studying the microscopic events of

traffic on the global bus, instead of having to use a statistical model to approximate the access pattern by the gateway controllers, one can run the procedures that generate the actual patterns. This can lead to insights which may otherwise be obscured.

It is hoped that through simulation using SHELL, a more optimal designs of INFOPLEX can be attained.

BIBLIOGGRAPHY

[Abraham] Abraham, M., "A Preliminary Design of the Control Structure of S.T.V.," unpublished lecture notes.

[Hoare] Hoare, C.A.R., "Monitors: An Operating System Structure Concept," Comm. of the ACM, Vol. 17, No. 10 (1974).

[Hsu 1] Hsu, Meichun, "A Preliminary Architectual Design For the Functional Hierarchy of the INFOPLEX Database Computer," Technical Report No. 5, Center for Information System Research, MIT, Cambridge.

[Hsu 2] Hsu, Meichun, "Functional Hierarchy Environment," unpublished memo. 13, No. 2 (1979).

[IBM] OS PL/1 Optimizing Compiler: Execution Logic, International Business Machine, INC Publication SY-0065.

[Madnick] Madnick, S.E., "The INFOPLEX Database Computer: Concepts and directions," in Proceedings of IEEE Computer Conference, (1977), pp.168-176.

[Wirth] Wirth, N., "Design and Implementation of Modula," Software - Practice and Experience, Vol. 7 (1977), pp.67-84.

APPENDIX A: LISTINGS OF PL/1 PROCEDURES

PL/1 Procedures:

AAHER
ADDQ
CONS
DEBUGR
EXECUTE
FINISH
GBER
GCER
QEVENT
SAHER
SEND
SHELL
SLEEP
STIMER
SVCER
SYNC
TERM
VPER
VPSTART
WAIT

SOURCE LISTING

```
NUMBER LEV NT                                                                    R

    10      0  AAHER: PROC (LOS,SVR,EVENT);                                      AAH00010
                                                                                 AAH00020
    30   1  0  DCL 1 LOS,                                                       AAH00030
                   70 LEVEL FIXED BIN,                                           LOS00010
                   70 SRQ,                                                       LOS00020
                      71 SIZE FIXED BIN,                                         LOS00030
                      71 MAX FIXED BIN,                                          LOS00040
                   70 DBB,                                                       LOS00050
                      71 SIZE FIXED BIN,                                         LOS00060
                      71 MAX FIXED BIN,                                          LOS00070
                   70 VPS,                                                       LOS00080
                      71 TABLE (      20),                                       LOS00090  1
                      80 BDSA FIXED BIN (31),       /* ADDR (BOTTOM DSA)       */VP 00010
                      80 TDSA FIXED BIN (31),       /* ADDR (TOP DSA)          */VP 00020
                      80 SAVESIZE FIXED BIN (31),   /* SIZE OF AREA TO BE SAVED*/VP 00030
                      80 SAFE PTR,                  /* -> 1ST SAVBLK           */VP 00040
                      80 PROCNAME CHAR (7) VAR,     /* NAME OF TOP LEVEL PROC  */VP 00050
                      80 STATUS CHAR (12) VAR,                                    VP 00060
                      80 MAIL PTR,                  /* CHAIN OF INCOMING MAIL  */VP 00070
                      80 WAIT,                                                    VP 00080
                         81 BOX FIXED BIN,          /* BOX AWAITING MAIL       */VP 00090
                         81 MSG PTR,                /* MSG IN WAIT.BOX         */VP 00100
                      80 VPID FIXED BIN,            /* INDEX IN THE VPST       */VP 00110
                      80 LEVEL FIXED BIN,           /* LEVEL                   */VP 00120
                      80 VTIME FIXED BIN (31)                                     VP 00130
                   ;                                                             AAH00030
300040   1  0  DCL 1 SVR,                                                        AAH00040
                   70 NEXT PTR,                                                  SVR00010
                   70 STIME FIXED BIN (31),                                      SVR00020
                   70 STIMEQ FIXED BIN (31),    /* ACCUMULATED STIME          */SVR00030
                   70 RTIME FIXED BIN (31),                                      SVR00040
                   70 RRATE FIXED BIN (15,7),   /* REAL CPU TICK RATE         */SVR00050
                   70 ID FIXED BIN                                               SVR00060
                   ;                                                             AAH00040
500050   1  0  DCL 1 EVENT,                                                      AAH00050
                   70 NEXT PTR,                 /* NEXT EVENT                 */ EVE00010
                   70 STIME FIXED BIN (31),                                      EVE00020
                   70 TYPE CHAR (12) VAR,                                        EVE00030
                   70 INDEX FIXED BIN (31),                                      EVE00040
                   70 PTR PTR                                                    EVE00050
                   ;                                                             AAH00050
700060   1  0  DCL (THISLOS,THISSVR) PTR EXTERNAL STATIC;                        AAH00060
700070   1  0  DCL STM FIXED BIN (31);                                          AAH00070
800010   1  0  DCL 1 DEBUG EXTERNAL STATIC,                                     DEB00010
                   2 SLEEP$ BIT(1) INIT ('0'B),                                  DEB00020
                   2 SAHER$ BIT(1) INIT ('0'B),                                  DEB00030
```

```
 NUMBER LEV NT                                                                  A

                     2 SCHEDULER$ BIT(1) INIT ('0'B),                          DEB00040
                     2 SHELL$ BIT(1) INIT ('0'B),                              DEB00050
                     2 AAHER$ BIT(1) INIT ('0'B);                              DEB00060
 900010   1  0   DCL PT_ADDR PTR;                                              PFA00010
                                                                               PFA00020
 900030   1  0   DCL 1 PF_ADDR BASED (PT_ADDR),                                PFA00030
                     2 VPID FIXED BIN,                                         PFA00040
                     2 BOXID FIXED BIN,                                        PFA00050
                     2 PTR PTR;                                                PFA00060
1000010   1  0   DCL PT_S PTR;                                                 PFS00010
                                                                               PFS00020
1000030   1  0   DCL 1 PF_S BASED (PT_S),                                      PFS00030
                     2 LEN FIXED BIN,                                          PFS00040
                     2 PTR PTR;                                                PFS00050
                                                                               AAH00110
1100010   1  0   DCL STIMER ENTRY RETURNS (FIXED BIN (31));                    STI00010
1200010   1  0   DCL QEVENT ENTRY (FIXED BIN, FIXED BIN (31), CHAR(*) VAR,     QEV00010
                                  FIXED BIN (31), PTR);                        QEV00020
1300010   1  0   DCL SVCER ENTRY (FIXED BIN, PTR);                            SVC00010
1300150   1  0   DCL (ADDR) BUILTIN;                                           AAH00150
                                                                               AAH00160
1300170   1  0       THISLOS = ADDR (LOS);                                     AAH00170
1300180   1  0       THISSVR = ADDR (SVR);                                     AAH00180
                                                                               AAH00190
1300200   1  0       IF (EVENT.TYPE = 'D') THEN DO;                            AAH00200
1300210   1  1          PT_ADDR = EVENT.PTR;                                   AAH00210
1300220   1  1          LOS.DBB.SIZE = LOS.DBB.SIZE - 1;                       AAH00220
1300230   1  1       END;                                                      AAH00230
1300240   1  0       ELSE DO; /* EVENT.TYPE = $ */                             AAH00240
1300250   1  1          PT_S = EVENT.PTR;                                      AAH00250
1300260   1  1          LOS.SRQ.SIZE = LOS.SRQ.SIZE - PF_S.LEN;                AAH00260
1300270   1  1          PT_ADDR = PF_S.PTR;                                    AAH00270
1300280   1  1          FREE PF_S;                                             AAH00280
1300290   1  1       END;                                                      AAH00290
                                                                               AAH00300
1300310   1  0       IF PF_ADDR.VPID = 0 THEN DO;                              AAH00310
1300320   1  1          CALL SVCER (PF_ADDR.BOXID,PF_ADDR.PTR);                AAH00320
1300330   1  1          FREE PF_ADDR;                                          AAH00330
1300340   1  1       END;                                                      AAH00340
1300350   1  0       ELSE DO;                                                  AAH00350
1300360   1  1          STM = STIMER;                                          AAH00360
1300370   1  1          CALL QEVENT (LOS.LEVEL*3+3,STM,'LOGMSG',0,PT_ADDR);    AAH00370
1300380   1  1       END;                                                      AAH00380
                                                                               AAH00390
1300400   1  0       STM = STIMER;                                             AAH00400
1300410   1  0       IF (DEBUG.AAHER$) THEN DO;                                AAH00410
1300420   1  1          PUT SKIP EDIT (' AAHER: TO = ',SVR.STIME,' T1 = ',     AAH00420
                           STM,' QT = ',SVR.STIMEQ)                            AAH00430
                           (A,F(11),A,F(11),A,F(11));                          AAH00440
```

PL/I OPTIMIZING COMPILER AAHER: PROC (LOS,SVR,EVENT);

R

```
NUMBER LEV NT

                                                                        AAH00450
                                                                        AAH00460
1300450   1  1       END;                                               AAH00470
1300460   1  0       SVR.STIMEQ = SVR.STIMEQ + STM - SVR.STIME;         AAH00480
1300470   1  0       SVR.STIME = STM;                                   AAH00490

1300490   1  0  END;
```

SOURCE LISTING

```
NUMBER LEV NT

    10      0  ADDQ: PROC (Q,PO);                                          ADD00010
                                                                           ADD00020
               /* Q -> HEAD OF QUEUE                                       ADD00030
                  PO -> OBJECT TO BE ADDED */                              ADD00040
                                                                           ADD00050
    60   1  0  DCL (Q,PQ,PQ2,PO,PT) PTR;                                   ADD00060
    70   1  0  DCL (ADDR,NULL) BUILTIN;                                    ADD00070
                                                                           ADD00080
               /* TB == FIRST PART OF SVR OR EVENT */                      ADD00090
   100   1  0  DCL 1 TB BASED (PT),                                        ADD00100
                    2 NEXT PTR,                                            ADD00110
                    2 STIME FIXED BIN (31);                                ADD00120
                                                                           ADD00130
   140   1  0      PQ = ADDR(Q);                                          ADD00140
   150   1  0      PQ2 = PQ->TB.NEXT;                                     ADD00150
   160   1  0      DO WHILE (PQ2 ^= NULL);                                ADD00160
   170   1  1         IF (PQ2->TB.STIME >= PO->TB.STIME) THEN GOTO FOUND; ADD00170
   180   1  1         PQ = PQ2;                                           ADD00180
   190   1  1         PQ2 = PQ->TB.NEXT;                                  ADD00190
   200   1  1      END;                                                   ADD00200
                                                                           ADD00210
   220   1  0  FOUND:                                                     ADD00220
               PO->TB.NEXT = PQ2;                                         ADD00230
   240   1  0      PQ->TB.NEXT = PO;                                      ADD00240
                                                                           ADD00250
   260   1  0  END ADDQ;                                                  ADD00260
                                                                           ADD00270
```

SOURCE LISTING

```
NUMBER LEV NT

    10      0  CONS: PROC (CAR, CDR) RETURNS (PTR);                    CON00010
                                                                        CON00020
               %INCLUDE LIST;*************************************************CON00030
100010   1   0  DCL LISTPT PTR;                                         LIS00010
                                                                        LIS00020
100030   1   0  DCL 1 LIST BASED (LISTPT),                             LIS00030
                      2 NEXT PTR,                                       LIS00040
                      2 THIS PTR;                                       LIS00050
                                                                        LIS00060
               **************                                           CON00030
200040   1   0  DCL (CAR, CDR) PTR;                                    CON00040
                                                                        CON00050
200060   1   0      ALLOCATE LIST;                                      CON00060
200070   1   0      LIST.NEXT = CDR;                                    CON00070
200080   1   0      LIST.THIS = CAR;                                    CON00080
200090   1   0      RETURN (LISTPT);                                    CON00090
                                                                        CON00100
200110   1   0  END CONS;                                              CON00110
```

SOURCE LISTING

```
NUMBER LEV NT

    10       0  EXECUTE: PROC (PROCNAME);                                    EXE00010
                                                                             EXE00020
    30   1   0  DCL PROCNAME CHAR(*) VAR;                                    EXE00030
    40   1   0  DCL TERM ENTRY;                                              EXE00040
                                                                             EXE00050
    60   1   0      IF PROCNAME = 'TERM' THEN DO; CALL TERM; RETURN; END;    EXE00060
                                                                             EXE00070
    80   1   0    PUT SKIP LIST (' ERROR: UNKNOWN PROCNAME. ',PROCNAME,      EXE00080
                                ' (EXECUTE).');                              EXE00090
   100   1   0      STOP;                                                    EXE00100
                                                                             EXE00110
   120   1   0  END EXECUTE;                                                 EXE00120
```

SOURCE LISTING

```
NUMBER LEV NT

    10      0  DEBUGR: PROC;                                                    DEB00010
                                                                                DEB00020
               %INCLUDE DEBUG;*****************************************************DEB00030
100010   1  0  DCL 1 DEBUG EXTERNAL STATIC,                                     DEB00010
                     2 SLEEP$ BIT(1) INIT ('0'B),                               DEB00020
                     2 SAHER$ BIT(1) INIT ('0'B),                               DEB00030
                     2 SCHEDULER$ BIT(1) INIT ('0'B),                           DEB00040
                     2 SHELL$ BIT(1) INIT ('0'B),                               DEB00050
                     2 AAHER$ BIT(1) INIT ('0'B);                               DEB00060
               **************                                                   DEB00030
                                                                                DEB00040
200050   1  0       DEBUG.SAHER$ = '1'B;                                        DEB00050
200060   1  0       DEBUG.SCHEDULER$ = '1'B;                                    DEB00060
200070   1  0       DEBUG.SHELL$ = '1'B;                                        DEB00070
200080   1  0       DEBUG.AAHER$ = '1'B;                                        DEB00080
                                                                                DEB00090
200100   1  0  END DEBUGR;                                                      DEB00100
```

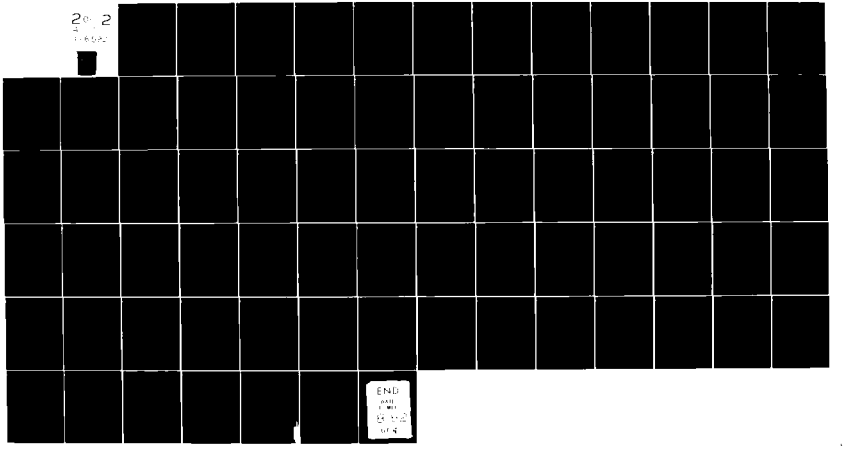
SOURCE LISTING

```
NUMBER LEV NT

    10      0  FINISH: PROC;                                                          FIN00010
                                                                                      FIN00020
               %INCLUDE VPX;*****************************************************FIN00030
100010   1  0  DCL THISVP PTR EXTERNAL STATIC;        /* -> CURRENT VP, SET BY VPER */VPX00010
                                                                                      VPX00020
100030   1  0  DCL 1 VP BASED (THISVP), %INCLUDE VP;*********************************VPX00030
                     80 BDSA FIXED BIN (31),        /* ADDR (BOTTOM DSA)          */VP 00010
                     80 TDSA FIXED BIN (31),        /* ADDR (TOP DSA)             */VP 00020
                     80 SAVESIZE FIXED BIN (31),    /* SIZE OF AREA TO BE SAVED   */VP 00030
                     80 SAFE PTR,                   /* -> 1ST SAVBLK              */VP 00040
                     80 PROCNAME CHAR (7) VAR,      /* NAME OF TOP LEVEL PROC     */VP 00050
                     80 STATUS CHAR (12) VAR,                                        VP 00060
                     80 MAIL PTR,                   /* CHAIN OF INCOMING MAIL     */VP 00070
                     80 WAIT,                                                        VP 00080
                        81 BOX FIXED BIN,           /* BOX AWAITING MAIL          */VP 00090
                        81 MSG PTR,                 /* MSG IN WAIT.BOX            */VP 00100
                     80 VPID FIXED BIN,             /* INDEX IN THE VPST          */VP 00110
                     80 LEVEL FIXED BIN,            /* LEVEL                      */VP 00120
                     80 VTIME FIXED BIN (31)                                         VP 00130
               ************************************    ;                           VPX00030
                                                                                      VPX00040
               *************                                                          FIN00030
                                                                                      FIN00040
               %INCLUDE QEVENT;*************************************************FIN00050
500010   1  0  DCL QEVENT ENTRY (FIXED BIN, FIXED BIN (31), CHAR(*) VAR,            QEV00010
                                FIXED BIN (31), PTR);                               QEV00020
               ****************                                                       FIN00050
500060   1  0  DCL NULL BUILTIN;                                                     FIN00060
500070   1  0  DCL SLEEP ENTRY;                                                     FIN00070
               %INCLUDE STIMER;*************************************************FIN00080
600010   1  0  DCL STIMER ENTRY RETURNS (FIXED BIN (31));                          STI00010
               *****************                                                    FIN00080
                                                                                   FIN00090
600100   1  0      VP.WAIT.BOX = -1;                                               FIN00100
600110   1  0      CALL QEVENT (VP.LEVEL*3+3,STIMER,'FINISH',VP.VPID,NULL);        FIN00110
600120   1  0      CALL SLEEP;                                                     FIN00120
                                                                                   FIN00130
600140   1  0  END;                                                               FIN00140
```

SOURCE LISTING

```
NUMBER LEV NT

    10      0  GBER: PROC (SVR,EVENT);                                          GBE00010
                                                                                GBE00020
    30   1  0  DCL 1 SVR, %INCLUDE SVR;*****************************************GBE00030
                   70 NEXT PTR,                                                 SVR00010
                   70 STIME FIXED BIN (31),                                     SVR00020
                   70 STIMEQ FIXED BIN (31),       /* ACCUMULATED STIME      */SVR00030
                   70 RTIME FIXED BIN (31),                                     SVR00040
                   70 RRATE FIXED BIN (15,7),      /* REAL CPU TICK RATE     */SVR00050
                   70 ID FIXED BIN                                              SVR00060
               ************************   ;                                    GBE00030
200040   1  0  DCL 1 EVENT, %INCLUDE EVENT;*************************************GBE00040
                   70 NEXT PTR,                    /* NEXT EVENT             */ EVE00010
                   70 STIME FIXED BIN (31),                                     EVE00020
                   70 TYPE CHAR (12) VAR,                                       EVE00030
                   70 INDEX FIXED BIN (31),                                     EVE00040
                   70 PTR PTR                                                   EVE00050
               ***************************   ;                                 GBE00040
400050   1  0  DCL STM FIXED BIN (31);                                          GBE00050
               %INCLUDE PARAMS;*************************************************GBE00060
500010   1  0  DCL 1 PARAMS EXTERNAL STATIC,                                    PAR00010
                   2 THRU_RATE FIXED BIN (31,7),                                PAR00020
                   2 DELAY_GB_GC FIXED BIN (31),                                PAR00030
                   2 DELAY_GC_GB FIXED BIN (31),                                PAR00040
                   2 TERMINALS FIXED BIN;                                       PAR00050
                                                                                PAR00060
               ****************                                                 GBE00060
600070   1  0  DCL STATS FILE EXTERNAL;                                         GBE00070
               %INCLUDE PFLEVEL;************************************************GBE00080
700010   1  0  DCL PT_LEVEL PTR;                                                PFL00010
                                                                                PFL00020
700030   1  0  DCL 1 PF_LEVEL BASED (PT_LEVEL),                                 PFL00030
                   2 LEVEL FIXED BIN,                                           PFL00040
                   2 PTR PTR;                                                   PFL00050
                                                                                PFL00060
               ******************                                               GBE00080
               %INCLUDE PFMSG;**************************************************GBE00090
800010   1  0  DCL PT_MSG PTR;                                                  PFM00010
                                                                                PFM00020
800030   1  0  DCL 1 PF_MSG BASED (PT_MSG),                                     PFM00030
                   2 LEN FIXED BIN,                                             PFM00040
                   2 TYPE CHAR (12) VAR,                                        PFM00050
                   2 PTR PTR;                                                   PFM00060
                                                                                PFM00070
               ****************                                                 GBE00090
                                                                                GBE00100
               %INCLUDE QEVENT;*************************************************GBE00110
```

AD-A116 592 ALFRED P SLOAN SCHOOL OF MANAGEMENT CAMBRIDGE MA CEN--ETC F/G 9/2
SHELL: A SIMULATOR FOR THE SOFTWARE TEST VEHICLE OF THE INFOPLE--ETC(U)
JAN 82 T TO N00039-81-C-0663
UNCLASSIFIED CISR-M010-8201-08 NL



```
NUMBER LEV NT

900010   1  0  DCL QEVENT ENTRY (FIXED BIN, FIXED BIN (31), CHAR(*) VAR,      QEV00010
                                FIXED BIN (31), PTR);                        QEV00020
              ****************                                               GBE00110
                                                                             GBE00120
900130   1  0      STM = EVENT.STIME + PARAMS.DELAY_GB_GC;                    GBE00130
900140   1  0      PT_LEVEL = EVENT.PTR;                                      GBE00140
900150   1  0      PT_MSG = PF_LEVEL.PTR;                                     GBE00150
900160   1  0      PUT FILE (STATS) SKIP EDIT (' GB: STIME = ',SVR.STIME,     GBE00160
                      ' MSGLEN = ',PF_MSG.LEN) (A,F(11),A,F(11));            GBE00170
900180   1  0      CALL QEVENT (PF_LEVEL.LEVEL*3+1,STM,'MSGIN',0,PT_MSG);     GBE00180
900190   1  0      FREE PF_LEVEL;                                             GBE00190
900200   1  0      SVR.STIME = EVENT.STIME;                                   GBE00200
                                                                             GBE00210
900220   1  0  END GBER;                                                      GBE00220
```

SOURCE LISTING

```
NUMBER LEV NT                                                                    R

    10      0  GCER: PROC (LOS,SVR,EVENT);                                      GCE00010
                                                                                GCE00020
                                                                                GCE00030
    30   1  0  DCL 1 LOS,                                                       LOS00010
                  70 LEVEL FIXED BIN,                                           LOS00020
                  70 SRQ,                                                       LOS00030
                     71 SIZE FIXED BIN,                                         LOS00040
                     71 MAX FIXED BIN,                                          LOS00050
                  70 DBB,                                                       LOS00060
                     71 SIZE FIXED BIN,                                         LOS00070
                     71 MAX FIXED BIN,                                          LOS00080
                  70 VPS,                                                       LOS00090  1
                     71 TABLE (      20),
                     80 BDSA FIXED BIN (31),     /* ADDR (BOTTOM DSA)          */VP 00010
                     80 TDSA FIXED BIN (31),     /* ADDR (TOP DSA)             */VP 00020
                     80 SAVESIZE FIXED BIN (31), /* SIZE OF AREA TO BE SAVED   */VP 00030
                     80 SAFE PTR,                /* -> 1ST SAVBLK              */VP 00040
                     80 PROCNAME CHAR (7) VAR,   /* NAME OF TOP LEVEL PROC     */VP 00050
                     80 STATUS CHAR (12) VAR,                                     VP 00060
                     80 MAIL PTR,                /* CHAIN OF INCOMING MAIL     */VP 00070
                     80 WAIT,                                                     VP 00080
                        81 BOX FIXED BIN,        /* BOX AWAITING MAIL          */VP 00090
                        81 MSG PTR,              /* MSG IN WAIT.BOX            */VP 00100
                     80 VPID FIXED BIN,          /* INDEX IN THE VPST          */VP 00110
                     80 LEVEL FIXED BIN,         /* LEVEL                      */VP 00120
                     80 VTIME FIXED BIN (31)                                      VP 00130
                  ;                                                             GCE00030
                                                                                GCE00040
300040   1  0  DCL 1 SVR,                                                       SVR00010
                  70 NEXT PTR,                                                  SVR00020
                  70 STIME FIXED BIN (31),
                  70 STIMEQ FIXED BIN (31),     /* ACCUMULATED STIME          */SVR00030
                  70 RTIME FIXED BIN (31),                                      SVR00040
                  70 RRATE FIXED BIN (15,7),    /* REAL CPU TICK RATE         */SVR00050
                  70 ID FIXED BIN                                               SVR00060
                  ;                                                             GCE00040
                                                                                GCE00050
500050   1  0  DCL 1 EVENT,                                                     */ EVE00010
                  70 NEXT PTR,                  /* NEXT EVENT                 EVE00020
                  70 STIME FIXED BIN (31),                                      EVE00030
                  70 TYPE CHAR (12) VAR,                                        EVE00040
                  70 INDEX FIXED BIN (31),                                      EVE00050
                  70 PTR PTR                                                    GCE00050
                  ;                                                             PAR00010
800010   1  0  DCL 1 PARAMS EXTERNAL STATIC,                                    PAR00020
                  2 THRU_RATE FIXED BIN (31,7),                                 PAR00030
                  2 DELAY_GB_GC FIXED BIN (31),                                 PAR00040
                  2 DELAY_GC_GB FIXED BIN (31),                                 PAR00050
                  2 TERMINALS FIXED BIN;
```

```
 NUMBER LEV NT                                                                  R

 800070   1  0  DCL STATS FILE EXTERNAL;                                        GCE00070
                                                                               GCE00080
 900010   1  0  DCL QEVENT ENTRY (FIXED BIN, FIXED BIN (31), CHAR(*) VAR,      QEV00010
                                 FIXED BIN (31), PTR);                         QEV00020
 900100   1  0  DCL MAX BUILTIN;                                               GCE00100
                                                                               GCE00110
 900120   1  0       PUT FILE (STATS) SKIP EDIT (' GC (',LOS.LEVEL,'): ',EVENT.TYPE,  GCE00120
                      ' DELAY = ',MAX(0,SVR.STIME-EVENT.STIME))                GCE00130
                      (A,F(2),A,A(6),A,F(11));                                 GCE00140
                                                                               GCE00150
 900160   1  0       IF (EVENT.TYPE = 'MSGIN') THEN CALL GCERIN;               GCE00160
 900170   1  0                                 ELSE CALL GCEROUT;              GCE00170
                                                                               GCE00180
 900190   1  0       RETURN;                                                   GCE00190
                                                                               GCE00200
                                                                               GCE00210
 900220   1  0  GCERIN: PROC;                                                  GCE00220
                                                                               GCE00230
1000010   2  0  DCL PT_MSG PTR;                                               PFM00010
                                                                               PFM00020
1000030   2  0  DCL 1 PF_MSG BASED (PT_MSG),                                   PFM00030
                    2 LEN FIXED BIN,                                           PFM00040
                    2 TYPE CHAR (12) VAR,                                      PFM00050
                    2 PTR PTR;                                                 PFM00060
1000250   2  0  DCL STM FIXED BIN (31);                                        GCE00250
                                                                               GCE00260
1000270   2  0       PT_MSG = EVENT.PTR;                                       GCE00270
1000280   2  0       IF (PF_MSG.TYPE = 'S') THEN DO;                           GCE00280
1000290   2  1          LOS.SRQ.SIZE = LOS.SRQ.SIZE + PF_MSG.LEN;              GCE00290
1000300   2  1          LOS.SRQ.MAX = MAX (LOS.SRQ.MAX, LOS.SRQ.SIZE);         GCE00300
1000310   2  1       END;                                                      GCE00310
1000320   2  0       ELSE DO; /* PF_MSG.TYPE = 'D' */                         GCE00320
1000330   2  1          LOS.DBB.SIZE = LOS.DBB.SIZE + 1;                       GCE00330
1000340   2  1          LOS.DBB.MAX = MAX (LOS.DBB.MAX, LOS.DBB.SIZE);         GCE00340
1000350   2  1       END;                                                      GCE00350
                                                                               GCE00360
1000370   2  0       STM = SVR.STIME + PARAMS.THRU_RATE * PF_MSG.LEN;          GCE00370
1000380   2  0       CALL QEVENT (LOS.LEVEL*3+2,STM,PF_MSG.TYPE,0,PF_MSG.PTR); GCE00380
1000390   2  0       FREE PF_MSG;                                              GCE00390
1000400   2  0       SVR.STIMEQ = SVR.STIMEQ + STM - SVR.STIME;                GCE00400
1000410   2  0       SVR.STIME = STM;                                          GCE00410
                                                                               GCE00420
1000430   2  0  END GCERIN;                                                    GCE00430
                                                                               GCE00440
                                                                               GCE00450
1000460   1  0  GCEROUT: PROC;                                                GCE00460
                                                                               GCE00470
1100010   2  0  DCL PT_LEVEL PTR;                                             PFL00010
                                                                               PFL00020
```

```
 NUMBER LEV NT                                                                    R

1100030   2  0  DCL 1 PF_LEVEL BASED (PT_LEVEL),                          PFL00030
                    2 LEVEL FIXED BIN,                                    PFL00040
                    2 PTR PTR;                                            PFL00050
1200010   2  0  DCL PT_MSG PTR;                                           PFM00010
                                                                          PFM00020
1200030   2  0  DCL 1 PF_MSG BASED (PT_MSG),                             PFM00030
                    2 LEN FIXED BIN,                                      PFM00040
                    2 TYPE CHAR (12) VAR,                                 PFM00050
                    2 PTR PTR;                                            PFM00060
1200500   2  0  DCL STM FIXED BIN (31);                                   GCE00500
                                                                          GCE00510
1200520   2  0       STM = SVR.STIME + PARAMS.DELAY_GC_GB;                GCE00520
1200530   2  0       CALL QEVENT (0,STM,'MSG',0,EVENT.PTR);               GCE00530
1200540   2  0       PT_LEVEL = EVENT.PTR;                                GCE00540
1200550   2  0       PT_MSG = PF_LEVEL.PTR;                               GCE00550
1200560   2  0       STM = PARAMS.THRU_RATE * PF_MSG.LEN;                 GCE00560
1200570   2  0       SVR.STIMEQ = SVR.STIMEQ + STM;                       GCE00570
1200580   2  0       SVR.STIME = SVR.STIME + STM;                         GCE00580
                                                                          GCE00590
1200600   2  0  END GCEROUT;                                             GCE00600
                                                                          GCE00610
                                                                          GCE00620
1200630   1  0  END GCER;                                                GCE00630
```

SOURCE LISTING

```
NUMBER LEV NT                                                                    R

    10       0  QEVENT: PROC (IQ,STIME,TYPE,INDEX,PTR);                        QEV00010
                                                                                QEV00020
100010   1   0  DCL SQUEUE (0:       6) PTR EXTERNAL STATIC;                   QS 00010  1
100020   1   0  DCL EQUEUE (0:       6) PTR EXTERNAL STATIC;                   QS 00020  1
                                                                                QEV00050
                                                                                QEV00060
100080   1   0  DCL EPT PTR;                                                    QEV00070
100070   1   0  DCL 1 EVENT BASED (EPT),                                      */ EVE00010
                    70 NEXT PTR,                  /* NEXT EVENT               EVE00020
                    70 STIME FIXED BIN (31),                                    EVE00030
                    70 TYPE CHAR (12) VAR,                                      EVE00040
                    70 INDEX FIXED BIN (31),                                    EVE00050
                    70 PTR PTR                                                  QEV00070
                  ;                                                             QEV00080
                                                                                QEV00090
200090   1   0  DCL IQ FIXED BIN;                                               QEV00100
200100   1   0  DCL STIME FIXED BIN (31);                                       QEV00110
200110   1   0  DCL TYPE CHAR(*) VAR;                                          QEV00120
200120   1   0  DCL INDEX FIXED BIN (31);                                      QEV00130
200130   1   0  DCL PTR PTR;                                                   QEV00140
                                                                                QEV00150
200150   1   0  DCL ADDQ ENTRY;                                                QEV00160
200160   1   0  DCL NULL BUILTIN;                                              QEV00170
                                                                                QEV00180
200180   1   0       ALLOCATE EVENT;                                          QEV00190
200190   1   0       EVENT.NEXT = NULL;                                       QEV00200
200200   1   0       EVENT.STIME = STIME;                                     QEV00210
200210   1   0       EVENT.TYPE = TYPE;                                       QEV00220
200220   1   0       EVENT.INDEX = INDEX;                                     QEV00230
200230   1   0       EVENT.PTR = PTR;                                         QEV00240
200240   1   0       CALL ADDQ (EQUEUE(IQ),EPT);                              QEV00250
                                                                                QEV00260
200260   1   0  END QEVENT;
```

SOURCE LISTING

```
NUMBER LEV NT                                                                    R

    10      0  SAHER: PROC (LOS,SVR,EVENT);                                      SAH00010
                                                                                 SAH00020
    30   1  0  DCL 1 LOS,                                                       SAH00030
                   70 LEVEL FIXED BIN,                                           LOS00010
                   70 SRQ,                                                       LOS00020
                      71 SIZE FIXED BIN,                                         LOS00030
                      71 MAX FIXED BIN,                                          LOS00040
                   70 DBB,                                                       LOS00050
                      71 SIZE FIXED BIN,                                         LOS00060
                      71 MAX FIXED BIN,                                          LOS00070
                   70 VPS,                                                       LOS00080
                      71 TABLE (      20),                                       LOS00090  1
                      80 BDSA FIXED BIN (31),       /* ADDR (BOTTOM DSA)       */VP 00010
                      80 TDSA FIXED BIN (31),       /* ADDR (TOP DSA)          */VP 00020
                      80 SAVESIZE FIXED BIN (31),   /* SIZE OF AREA TO BE SAVED*/VP 00030
                      80 SAFE PTR,                  /* -> 1ST SAVBLK           */VP 00040
                      80 PROCNAME CHAR (7) VAR,     /* NAME OF TOP LEVEL PROC  */VP 00050
                      80 STATUS CHAR (12) VAR,                                    VP 00060
                      80 MAIL PTR,                  /* CHAIN OF INCOMING MAIL  */VP 00070
                      80 WAIT,                                                    VP 00080
                         81 BOX FIXED BIN,          /* BOX AWAITING MAIL       */VP 00090
                         81 MSG PTR,                /* MSG IN WAIT.BOX         */VP 00100
                      80 VPID FIXED BIN,            /* INDEX IN THE VPST       */VP 00110
                      80 LEVEL FIXED BIN,           /* LEVEL                   */VP 00120
                      80 VTIME FIXED BIN (31)                                     VP 00130
                   ;                                                             SAH00030
300040   1  0  DCL 1 SVR,                                                        SAH00040
                   70 NEXT PTR,                                                  SVR00010
                   70 STIME FIXED BIN (31),                                      SVR00020
                   70 STIMEQ FIXED BIN (31),    /* ACCUMULATED STIME          */SVR00030
                   70 RTIME FIXED BIN (31),                                     SVR00040
                   70 RRATE FIXED BIN (15,7),   /* REAL CPU TICK RATE         */SVR00050
                   70 ID FIXED BIN                                               SVR00060
                   ;                                                             SAH00040
500050   1  0  DCL 1 EVENT,                                                      SAH00050
                   70 NEXT PTR,                 /* NEXT EVENT                 */ EVE00010
                   70 STIME FIXED BIN (31),                                      EVE00020
                   70 TYPE CHAR (12) VAR,                                        EVE00030
                   70 INDEX FIXED BIN (31),                                      EVE00040
                   70 PTR PTR                                                    EVE00050
                  ;                                                             SAH00050
                                                                                 SAH00060
700070   1  0  DCL (THISLOS,THISSVR) PTR EXTERNAL STATIC;                        SAH00070
700080   1  0  DCL STM FIXED BIN (31);                                          SAH00080
800010   1  0  DCL 1 PARAMS EXTERNAL STATIC,                                    PAR00010
                   2 THRU_RATE FIXED BIN (31,7),                                 PAR00020
```

```
NUMBER LEV NT                                                                    R

                      2 DELAY_GB_GC FIXED BIN (31),                              PAR00030
                      2 DELAY_GC_GB FIXED BIN (31),                              PAR00040
                      2 TERMINALS FIXED BIN;                                     PAR00050
 900010   1  0  DCL 1 DEBUG EXTERNAL STATIC,                                     DEB00010
                      2 SLEEP$ BIT(1) INIT ('0'B),                               DEB00020
                      2 SAHER$ BIT(1) INIT ('0'B),                               DEB00030
                      2 SCHEDULER$ BIT(1) INIT ('0'B),                           DEB00040
                      2 SHELL$ BIT(1) INIT ('0'B),                               DEB00050
                      2 AAHER$ BIT(1) INIT ('0'B);                               DEB00060
                                                                                 SAH00110
 900120   1  0  DCL (ADDR,NULL) BUILTIN;                                         SAH00120
1000010   1  0  DCL STIMER ENTRY RETURNS (FIXED BIN (31));                       STI00010
1000140   1  0  DCL VPER ENTRY;                                                  SAH00140
                                                                                 SAH00150
1000160   1  0      THISLOS = ADDR (LOS);                                       SAH00160
1000170   1  0      THISSVR = ADDR (SVR);                                       SAH00170
                                                                                 SAH00180
1000190   1  0      IF (DEBUG.SAHER$) THEN DO;                                  SAH00190
1000200   1  1         PUT SKIP EDIT (' SAHER: LEVEL = ',LOS.LEVEL,             SAH00200
                          ' S = (',SVR.ID,SVR.STIME,')',                         SAH00210
                          ' E = (',EVENT.TYPE,EVENT.STIME,')')                   SAH00220
                         (A,F(2), A,F(2),F(11),A, A,A(12),F(11),A);              SAH00230
1000240   1  1      END;                                                        SAH00240
                                                                                 SAH00250
1000260   1  0      IF (EVENT.TYPE = 'LOGMSG') THEN DO;                         SAH00260
1000270   1  1         CALL MAILMAN (LOS.VPS, EVENT.PTR);                       SAH00270
1000280   1  1      END;                                                        SAH00280
1000290   1  0      ELSE IF (EVENT.TYPE = 'SYNC') THEN DO;                      SAH00290
1000300   1  1         IF (LOS.VPS.TABLE(EVENT.INDEX).STATUS$ = 'RUNNING') THEN SAH00300
                          LOS.VPS.TABLE(EVENT.INDEX).STATUS$ = 'RUNNABLE';       SAH00310
                          /* DO NOT CHANGE NASCENT */                            SAH00320
1000330   1  1      END;                                                        SAH00330
1000340   1  0      ELSE IF (EVENT.TYPE = 'WAIT') THEN DO;                      SAH00340
                        /* WAIT.BOX WAS SET EARLIER */                           SAH00350
1000360   1  1         LOS.VPS.TABLE(EVENT.INDEX).STATUS = 'BLOCKED';           SAH00360
1000370   1  1      END;                                                        SAH00370
1000380   1  0      ELSE IF (EVENT.TYPE = 'FINISH') THEN DO;                    SAH00380
                        /* WAIT.BOX WAS SET EARLIER */                           SAH00390
1000400   1  1         LOS.VPS.TABLE(EVENT.INDEX).STATUS$ = 'VOID';             SAH00400
1000410   1  1      END;                                                        SAH00410
1000420   1  0      ELSE DO; /* EVENT.TYPE = INIT */                            SAH00420
1000430   1  1         CALL INITER (LOS.VPS);                                   SAH00430
1000440   1  1         RETURN; /* DO NOT SCHEDULE */                            SAH00440
1000450   1  1      END;                                                        SAH00450
                                                                                 SAH00460
1000470   1  0      CALL CHECK_MAIL (LOS.VPS);                                  SAH00470
1000480   1  0      CALL SCHEDULER (LOS.VPS);                                   SAH00480
                                                                                 SAH00490
1000500   1  0      STM = STIMER;                                               SAH00500
```

```
 NUMBER LEV NT                                                                                    R

1000510   1  0      SVR.STIMEQ = SVR.STIMEQ + STM - SVR.STIME;                                    SAH00510
1000520   1  0      SVR.STIME = STM;                                                              SAH00520
                                                                                                 SAH00530
1000540   1  0      RETURN;                                                                       SAH00540
                                                                                                 SAH00550
                                                                                                 SAH00560
1000570   1  0  MAILMAN: PROC (VPS, PT_ADDR);                                                   SAH00570
                                                                                                 SAH00580
1000590   2  0  DCL 1 VPS, 2 TABLE (      20),                                                  SAH00590  1
                    80 BDSA FIXED BIN (31),        /* ADDR (BOTTOM DSA)                      */VP 00010
                    80 TDSA FIXED BIN (31),        /* ADDR (TOP DSA)                         */VP 00020
                    80 SAVESIZE FIXED BIN (31),    /* SIZE OF AREA TO BE SAVED               */VP 00030
                    80 SAFE PTR,                   /* -> 1ST SAVBLK                          */VP 00040
                    80 PROCNAME CHAR (7) VAR,      /* NAME OF TOP LEVEL PROC                 */VP 00050
                    80 STATUS CHAR (12) VAR,                                                     VP 00060
                    80 MAIL PTR,                   /* CHAIN OF INCOMING MAIL                 */VP 00070
                    80 WAIT,                                                                     VP 00080
                       81 BOX FIXED BIN,           /* BOX AWAITING MAIL                      */VP 00090
                       81 MSG PTR,                 /* MSG IN WAIT.BOX                        */VP 00100
                    80 VPID FIXED BIN,             /* INDEX IN THE VPST                      */VP 00110
                    80 LEVEL FIXED BIN,            /* LEVEL                                  */VP 00120
                    80 VTIME FIXED BIN (31)                                                      VP 00130
                 ;                                                                               SAH00590
1200010   2  0  DCL PT_ADDR PTR;                                                                 PFA00010
                                                                                                 PFA00020
1200030   2  0  DCL 1 PF_ADDR BASED (PT_ADDR),                                                   PFA00030
                    2 VPID FIXED BIN,                                                            PFA00040
                    2 BOXID FIXED BIN,                                                           PFA00050
                    2 PTR PTR;                                                                   PFA00060
1300010   2  0  DCL BOXPT PTR;                                                                  BOX00010
                                                                                                 BOX00020
1300030   2  0  DCL 1 BOX BASED (BOXPT),                                                        BOX00030
                    2 NEXT PTR,                                                                  BOX00040
                    2 ID FIXED BIN,                                                              BOX00050
                    2 LIST PTR;                                                                  BOX00060
1300620   2  0  DCL VPID FIXED BIN;                                                            SAH00620
                                                                                                 SAH00630
1400010   2  0  DCL CONS ENTRY (PTR, PTR) RETURNS (PTR);                                       CON00010
                                                                                                 SAH00650
1400660   2  0      VPID = PF_ADDR.VPID;                                                        SAH00660
1400670   2  0      IF (VPS.TABLE(VPID).STATUS = 'VOID') THEN DO;                               SAH00670
1400680   2  1         PUT SKIP EDIT ( 'WARNING: ILLEGAL ADDRESS ',LOS.LEVEL,                   SAH00680
                          '/',VPID) (A,F(2),A,F(2));                                             SAH00690
1400700   2  1         FREE PF_ADDR;                                                           SAH00700
                     /* CANNOT FREE VARIABLE FORMAT MSG */                                     SAH00710
1400720   2  1         RETURN;                                                                 SAH00720
1400730   2  1      END;                                                                        SAH00730
                                                                                                 SAH00740
1400750   2  0      BOXPT = VPS.TABLE(VPID).MAIL;                                               SAH00750
```

```
 NUMBER LEV NT                                                                                    R

1400760   2  0      DO WHILE (BOXPT ^= NULL);                                                    SAH00760
1400770   2  1         IF BOX.ID = PF_ADDR.BOXID THEN GOTO FOUND;                                SAH00770
1400780   2  1         BOXPT = BOX.NEXT;                                                         SAH00780
1400790   2  1      END;                                                                         SAH00790
                                                                                                SAH00800
                      /* CREATE NEW BOX */                                                      SAH00810
1400820   2  0      ALLOCATE BOX;                                                              SAH00820
1400830   2  0      BOX.ID = PF_ADDR.BOXID;                                                    SAH00830
1400840   2  0      BOX.LIST = NULL;                                                           SAH00840
1400850   2  0      BOX.NEXT = VPS.TABLE(VPID).MAIL;                                           SAH00850
1400860   2  0      VPS.TABLE(VPID).MAIL = BOXPT;                                              SAH00860
                                                                                                SAH00870
1400880   2  0  FOUND:                                                                         SAH00880
                      BOX.LIST = CONS(PF_ADDR.PTR, BOX.LIST);                                   SAH00890
1400900   2  0      FREE PF_ADDR;                                                              SAH00900
                                                                                                SAH00910
1400920   2  0  END MAILMAN;                                                                   SAH00920
                                                                                                SAH00930
                                                                                                SAH00940
1400950   1  0  CHECK_MAIL: PROC (VPS);                                                        SAH00950
                                                                                                SAH00960
1400970   2  0  DCL 1 VPS, 2 TABLE (      20),                                                SAH00970  1
                        80 BDSA FIXED BIN (31),       /* ADDR (BOTTOM DSA)                  */VP 00010
                        80 TDSA FIXED BIN (31),       /* ADDR (TOP DSA)                     */VP 00020
                        80 SAVESIZE FIXED BIN (31),   /* SIZE OF AREA TO BE SAVED           */VP 00030
                        80 SAFE PTR,                  /* -> 1ST SAVBLK                      */VP 00040
                        80 PROCNAME CHAR (7) VAR,     /* NAME OF TOP LEVEL PROC             */VP 00050
                        80 STATUS CHAR (12) VAR,                                                VP 00060
                        80 MAIL PTR,                  /* CHAIN OF INCOMING MAIL             */VP 00070
                        80 WAIT,                                                                VP 00080
                           81 BOX FIXED BIN,          /* BOX AWAITING MAIL                  */VP 00090
                           81 MSG PTR,                /* MSG IN WAIT.BOX                    */VP 00100
                        80 VPID FIXED BIN,            /* INDEX IN THE VPST                  */VP 00110
                        80 LEVEL FIXED BIN,           /* LEVEL                              */VP 00120
                        80 VTIME FIXED BIN (31)                                                 VP 00130
                     ;                                                                          SAH00970
                                                                                                SAH00980
1600010   2  0  DCL BOXPT PTR;                                                                 BOX00010
                                                                                                BOX00020
1600030   2  0  DCL 1 BOX BASED (BOXPT),                                                       BOX00030
                     2 NEXT PTR,                                                                BOX00040
                     2 ID FIXED BIN,                                                            BOX00050
                     2 LIST PTR;                                                                BOX00060
1700010   2  0  DCL LISTPT PTR;                                                                LIS00010
                                                                                                LIS00020
1700030   2  0  DCL 1 LIST BASED (LISTPT),                                                     LIS00030
                     2 NEXT PTR,                                                                LIS00040
                     2 THIS PTR;                                                                LIS00050
1701010   2  0  DCL IV FIXED BIN;                                                             SAH01010
```

```
 NUMBER LEV NT                                                                                    R

1701020   2  0  DCL PT PTR;                                                                      SAH01020
                                                                                                 SAH01030
1701040   2  0       DO IV = 1 TO       20;                                                      SAH01040  1
1701050   2  1          IF VPS.TABLE(IV).STATUS = 'BLOCKED' THEN DO;                             SAH01050
1701060   2  2             BOXPT = VPS.TABLE(IV).MAIL;                                           SAH01060
1701070   2  2             DO WHILE (BOXPT ^= NULL);                                             SAH01070
1701080   2  3                IF BOX.ID = VPS.TABLE(IV).WAIT.BOX THEN DO;                        SAH01080
1701090   2  4                   IF BOX.LIST ^= NULL THEN DO;                                    SAH01090
1701100   2  5                      LISTPT = BOX.LIST;                                           SAH01100
1701110   2  5                      PT = LIST.THIS;                                              SAH01110
1701120   2  5                      BOX.LIST = LIST.NEXT;                                        SAH01120
1701130   2  5                      FREE LIST;                                                   SAH01130
1701140   2  5                      VPS.TABLE(IV).WAIT.MSG = PT;                                 SAH01140
1701150   2  5                      VPS.TABLE(IV).STATUS = 'RUNNABLE';                           SAH01150
1701160   2  5                      GOTO NEXT_VP;                                                SAH01160
1701170   2  5                   END;                                                            SAH01170
1701180   2  4                END;                                                               SAH01180
1701190   2  3                BOXPT = BOX.NEXT;                                                  SAH01190
1701200   2  3             END;                                                                  SAH01200
1701210   2  2          END;                                                                     SAH01210
1701220   2  1       NEXT_VP: END;                                                               SAH01220
                                                                                                 SAH01230
1701240   2  0  END CHECK_MAIL;                                                                  SAH01240
                                                                                                 SAH01250
                                                                                                 SAH01260
1701270   1  0  SCHEDULER: PROC (VPS);                                                          SAH01270
                                                                                                 SAH01280
1701290   2  0  DCL 1 VPS, 2 TABLE (      20),                                                  SAH01290  1
                    80 BDSA FIXED BIN (31),       /* ADDR (BOTTOM DSA)          */VP 00010
                    80 TDSA FIXED BIN (31),       /* ADDR (TOP DSA)             */VP 00020
                    80 SAVESIZE FIXED BIN (31),   /* SIZE OF AREA TO BE SAVED   */VP 00030
                    80 SAFE PTR,                  /* -> 1ST SAVBLK              */VP 00040
                    80 PROCNAME CHAR (7) VAR,     /* NAME OF TOP LEVEL PROC     */VP 00050
                    80 STATUS CHAR (12) VAR,                                        VP 00060
                    80 MAIL PTR,                  /* CHAIN OF INCOMING MAIL     */VP 00070
                    80 WAIT,                                                        VP 00080
                       81 BOX FIXED BIN,          /* BOX AWAITING MAIL          */VP 00090
                       81 MSG PTR,                /* MSG IN WAIT.BOX            */VP 00100
                    80 VPID FIXED BIN,            /* INDEX IN THE VPST          */VP 00110
                    80 LEVEL FIXED BIN,           /* LEVEL                      */VP 00120
                    80 VTIME FIXED BIN (31)                                         VP 00130
                ;                                                                   SAH01290
                                                                                    SAH01300
1801310   2  0  DCL (VTM,VTM0) FIXED BIN (31);                                      SAH01310
1801320   2  0  DCL YOUNGEST FIXED BIN;                                             SAH01320
1801330   2  0  DCL IV FIXED BIN;                                                   SAH01330
                                                                                    SAH01340
1801350   2  0       YOUNGEST = 0;                                                  SAH01350
1801360   2  0       VTM0 = 2147483647;                                             SAH01360
```

```
 NUMBER LEV NT                                                                                    R

                                                                                                  SAH01370
1801380   2  0      DO IV = 1 TO       20;                                                        SAH01380  1
1801390   2  1         IF (VPS.TABLE(IV).STATUS = 'RUNNABLE') |                                   SAH01390
                        (VPS.TABLE(IV).STATUS = 'NASCENT') THEN DO;                               SAH01400
1801410   2  2            VTM = VPS.TABLE(IV).VTIME;                                             SAH01410
1801420   2  2            IF VTM < VTMO THEN DO;                                                 SAH01420
1801430   2  3               YOUNGEST = IV;                                                      SAH01430
1801440   2  3               VTMO = VTM;                                                         SAH01440
1801450   2  3            END;                                                                   SAH01450
1801460   2  2         END;                                                                      SAH01460
1801470   2  1      END;                                                                         SAH01470
                                                                                                  SAH01480
1801490   2  0      IF (DEBUG.SCHEDULER$) THEN DO;                                             SAH01490
1801500   2  1         PUT SKIP EDIT (' SCHEDULER: YOUNGEST = ',YOUNGEST) (A,F(3));            SAH01500
1801510   2  1      END;                                                                         SAH01510
                                                                                                  SAH01520
1801530   2  0      IF (YOUNGEST > 0) THEN DO;                                                 SAH01530
1801540   2  1         CALL VPER (VPS.TABLE(YOUNGEST));                                        SAH01540
1801550   2  1      END;                                                                         SAH01550
                                                                                                  SAH01560
1801570   2  0   END SCHEDULER;                                                                 SAH01570
                                                                                                  SAH01580
                                                                                                  SAH01590
1801600   1  0   INITER: PROC (VPS);                                                           SAH01600
                                                                                                  SAH01610
1801620   2  0   DCL 1 VPS, 2 TABLE (      20),                                               SAH01620  1
                        80 BDSA FIXED BIN (31),       /* ADDR (BOTTOM DSA)          */VP 00010
                        80 TDSA FIXED BIN (31),       /* ADDR (TOP DSA)             */VP 00020
                        80 SAVESIZE FIXED BIN (31),   /* SIZE OF AREA TO BE SAVED   */VP 00030
                        80 SAFE PTR,                  /* -> 1ST SAVBLK              */VP 00040
                        80 PROCNAME CHAR (7) VAR,     /* NAME OF TOP LEVEL PROC     */VP 00050
                        80 STATUS CHAR (12) VAR,                                        VP 00060
                        80 MAIL PTR,                  /* CHAIN OF INCOMING MAIL     */VP 00070
                        80 WAIT,                                                        VP 00080
                           81 BOX FIXED BIN,          /* BOX AWAITING MAIL          */VP 00090
                           81 MSG PTR,                /* MSG IN WAIT.BOX            */VP 00100
                        80 VPID FIXED BIN,            /* INDEX IN THE VPST          */VP 00110
                        80 LEVEL FIXED BIN,           /* LEVEL                      */VP 00120
                        80 VTIME FIXED BIN (31)                                         VP 00130
                     ;                                                                  SAH01620
2000010   2  0   DCL 1 PARAMS EXTERNAL STATIC,                                           PAR00010
                       2 THRU_RATE FIXED BIN (31,7),                                    PAR00020
                       2 DELAY_GB_GC FIXED BIN (31),                                    PAR00030
                       2 DELAY_GC_GB FIXED BIN (31),                                    PAR00040
                       2 TERMINALS FIXED BIN;                                           PAR00050
2001640   2  0   DCL IV FIXED BIN;                                                       SAH01640
2100010   2  0   DCL MSGLEN FIXED BIN;                                                  MSG00010
2100020   2  0   DCL MSGPT PTR;                                                         MSG00020
2100030   2  0   DCL 1 MSG BASED (MSGPT),                                               MSG00030
```

```
 NUMBER LEV NT                                                                  R

                      2 LEN FIXED BIN,                                          MSG00040
                      2 STR CHAR (MSGLEN REFER (LEN));                          MSG00050
2200010   2  0  DCL VPSTART ENTRY (/* VP */, CHAR(*) VAR, PTR);                 VPS00010
                                                                                SAH01670
2201680   2  0       MSGLEN = 16;                                              SAH01680
2201690   2  0       DO IV = 1 TO PARAMS.TERMINALS;                            SAH01690
2201700   2  1          ALLOCATE MSG;                                          SAH01700
2201710   2  1          MSG.STR = 'SET UP BY INITER';                          SAH01710
2201720   2  1          CALL VPSTART (VPS.TABLE(IV),'TERM',MSGPT);             SAH01720
2201730   2  1       END;                                                      SAH01730
                                                                                SAH01740
2201750   2  0  END INITER;                                                    SAH01750
                                                                                SAH01760
                                                                                SAH01770
2201780   1  0  END SAHER;                                                     SAH01780
```

SOURCE LISTING

```
NUMBER LEV NT

    10      0  SEND: PROC (LEVEL,VPID,BOXID,TYPE,LEN,MPT);                    SEN00010
                                                                               SEN00020
    30   1  0  DCL (LEVEL,VPID,BOXID,LEN) FIXED BIN;                          SEN00030
    40   1  0  DCL (TYPE) CHAR(*) VAR;                                        SEN00040
    50   1  0  DCL (MPT,PT) PTR;                                              SEN00050
               %INCLUDE PFADDR;*************************************************SEN00060
100010   1  0  DCL PT_ADDR PTR;                                               PFA00010
                                                                               PFA00020
100030   1  0  DCL 1 PF_ADDR BASED (PT_ADDR),                                 PFA00030
                     2 VPID FIXED BIN,                                         PFA00040
                     2 BOXID FIXED BIN,                                        PFA00050
                     2 PTR PTR;                                                PFA00060
                                                                               PFA00070
               ****************                                                SEN00060
               %INCLUDE PFS;****************************************************SEN00070
300010   1  0  DCL PT_S PTR;                                                  PFS00010
                                                                               PFS00020
300030   1  0  DCL 1 PF_S BASED (PT_S),                                       PFS00030
                     2 LEN FIXED BIN,                                          PFS00040
                     2 PTR PTR;                                                PFS00050
                                                                               PFS00060
               *************                                                   SEN00070
               %INCLUDE PFMSG;**************************************************SEN00080
400010   1  0  DCL PT_MSG PTR;                                                PFM00010
                                                                               PFM00020
400030   1  0  DCL 1 PF_MSG BASED (PT_MSG),                                   PFM00030
                     2 LEN FIXED BIN,                                          PFM00040
                     2 TYPE CHAR (12) VAR,                                     PFM00050
                     2 PTR PTR;                                                PFM00060
                                                                               PFM00070
               ***************                                                 SEN00080
               %INCLUDE PFLEVEL;************************************************SEN00090
500010   1  0  DCL PT_LEVEL PTR;                                              PFL00010
                                                                               PFL00020
500030   1  0  DCL 1 PF_LEVEL BASED (PT_LEVEL),                               PFL00030
                     2 LEVEL FIXED BIN,                                        PFL00040
                     2 PTR PTR;                                                PFL00050
                                                                               PFL00060
               *****************                                               SEN00090
               %INCLUDE VPX;****************************************************SEN00100
600010   1  0  DCL THISVP PTR EXTERNAL STATIC;        /* -> CURRENT VP, SET BY VPER */VPX00010
                                                                               VPX00020
600030   1  0  DCL 1 VP BASED (THISVP), %INCLUDE VP;*****************************VPX00030
                     80 BDSA FIXED BIN (31),       /* ADDR (BOTTOM DSA)          */VP 00010
                     80 TDSA FIXED BIN (31),       /* ADDR (TOP DSA)             */VP 00020
                     80 SAVESIZE FIXED BIN (31),   /* SIZE OF AREA TO BE SAVED   */VP 00030
```

```
 NUMBER LEV NT

                        80 SAFE PTR,                    /* -> 1ST SAVBLK            */VP 00040
                        80 PROCNAME CHAR (7) VAR,       /* NAME OF TOP LEVEL PROC   */VP 00050
                        80 STATUS CHAR (12) VAR,                                      VP 00060
                        80 MAIL PTR,                    /* CHAIN OF INCOMING MAIL   */VP 00070
                        80 WAIT,                                                      VP 00080
                           81 BOX FIXED BIN,            /* BOX AWAITING MAIL        */VP 00090
                           81 MSG PTR,                  /* MSG IN WAIT.BOX          */VP 00100
                        80 VPID FIXED BIN,              /* INDEX IN THE VPST        */VP 00110
                        80 LEVEL FIXED BIN,             /* LEVEL                    */VP 00120
                        80 VTIME FIXED BIN (31)                                       VP 00130
                     ************************************    ;                        VPX00030
                                                                                      VPX00040
                     *************                                                    SEN00100
                                                                                      SEN00110
                     %INCLUDE QEVENT;*****************************************************SEN00120
  900010   1  0      DCL QEVENT ENTRY (FIXED BIN, FIXED BIN (31), CHAR(*) VAR,       QEV00010
                                 FIXED BIN (31), PTR);                                QEV00020
                     ***************                                                  SEN00120
                     %INCLUDE STIMER;*****************************************************SEN00130
 1000010   1  0      DCL STIMER ENTRY RETURNS (FIXED BIN (31));                      STI00010
                     ***************                                                  SEN00130
                     %INCLUDE SVCER;******************************************************SEN00140
 1100010   1  0      DCL SVCER ENTRY (FIXED BIN, PTR);                               SVC00010
                     ***************                                                  SEN00140
 1100150   1  0      DCL SYNC ENTRY;                                                 SEN00150
                                                                                      SEN00160
 1100170   1  0           IF (LEVEL = VP.LEVEL) & (VPID = 0) THEN DO;                SEN00170
 1100180   1  1              CALL SYNC;                                              SEN00180
 1100190   1  1              CALL SVCER (BOXID,MPT);                                 SEN00190
 1100200   1  1              RETURN;                                                 SEN00200
 1100210   1  1           END;                                                       SEN00210
                                                                                      SEN00220
                                                                                      SEN00230
 1100240   1  0           ALLOCATE PF_ADDR;                                          SEN00240
 1100250   1  0           PF_ADDR.BOXID = BOXID;                                     SEN00250
 1100260   1  0           PF_ADDR.VPID = VPID;                                       SEN00260
 1100270   1  0           PF_ADDR.PTR = MPT;                                         SEN00270
                                                                                      SEN00280
 1100290   1  0           IF (LEVEL = VP.LEVEL) THEN DO;                             SEN00290
 1100300   1  1              CALL QEVENT (VP.LEVEL*3+3,STIMER,'LOGMSG',0,PT_ADDR);   SEN00300
 1100310   1  1           END;                                                       SEN00310
                                                                                      SEN00320
 1100330   1  0           ELSE DO; /* LEVEL ^= VP.LEVEL */                           SEN00330
 1100340   1  1              PT = PT_ADDR;                                           SEN00340
 1100350   1  1              IF (TYPE = 'S') THEN DO;                                SEN00350
 1100360   1  2                 ALLOCATE PF_S;                                       SEN00360
 1100370   1  2                 PF_S.LEN = LEN;                                      SEN00370
 1100380   1  2                 PF_S.PTR = PT_ADDR;                                  SEN00380
 1100390   1  2                 PT = PT_S;                                           SEN00390
```

```
 NUMBER LEV NT

1100400   1  2          END;                                                    SEN00400
1100410   1  1          ALLOCATE PF_MSG;                                        SEN00410
1100420   1  1          PF_MSG.LEN = LEN+2;                                     SEN00420
1100430   1  1          PF_MSG.TYPE = TYPE;                                     SEN00430
1100440   1  1          PF_MSG.PTR = PT;                                        SEN00440
1100450   1  1          ALLOCATE PF_LEVEL;                                      SEN00450
1100460   1  1          PF_LEVEL.LEVEL = LEVEL;                                 SEN00460
1100470   1  1          PF_LEVEL.PTR = PT_MSG;                                  SEN00470
1100480   1  1          CALL QEVENT (VP.LEVEL+3+1,STIMER,'MSGOUT',0,PT_LEVEL);  SEN00480
1100490   1  1       END;                                                       SEN00490
                                                                              SEN00500
1100510   1  0  END SEND;                                                      SEN00510
```

SOURCE LISTING

```
NUMBER LEV NT                                                                    R

    10      0  SHELL: PROC OPTIONS (MAIN);                                      SHE00010
                                                                                SHE00020
    30   1  0  DCL PLIXOPT CHAR(40) VAR EXTERNAL STATIC                         SHE00030
                    INIT ('ISASIZE(-50K)');                                     SHE00040
                                                                                SHE00050
100010   1  0  DCL SQUEUE (0:       6) PTR EXTERNAL STATIC;                     QS 00010  1
100020   1  0  DCL EQUEUE (0:       6) PTR EXTERNAL STATIC;                     QS 00020  1
100080   1  0  DCL 1 LOSES (0:       1) STATIC,                                 SHE00080  1
                  70 LEVEL FIXED BIN,                                           LOS00010
                  70 SRQ,                                                       LOS00020
                     71 SIZE FIXED BIN,                                         LOS00030
                     71 MAX FIXED BIN,                                          LOS00040
                  70 DBB,                                                       LOS00050
                     71 SIZE FIXED BIN,                                         LOS00060
                     71 MAX FIXED BIN,                                          LOS00070
                  70 VPS,                                                       LOS00080
                     71 TABLE (      20),                                       LOS00090  1
                     80 BDSA FIXED BIN (31),      /* ADDR (BOTTOM DSA)        */VP 00010
                     80 TDSA FIXED BIN (31),      /* ADDR (TOP DSA)           */VP 00020
                     80 SAVESIZE FIXED BIN (31),  /* SIZE OF AREA TO BE SAVED */VP 00030
                     80 SAFE PTR,                 /* -> 1ST SAVBLK            */VP 00040
                     80 PROCNAME CHAR (7) VAR,    /* NAME OF TOP LEVEL PROC   */VP 00050
                     80 STATUS CHAR (12) VAR,                                     VP 00060
                     80 MAIL PTR,                 /* CHAIN OF INCOMING MAIL   */VP 00070
                     80 WAIT,                                                     VP 00080
                        81 BOX FIXED BIN,         /* BOX AWAITING MAIL        */VP 00090
                        81 MSG PTR,               /* MSG IN WAIT.BOX          */VP 00100
                     80 VPID FIXED BIN,           /* INDEX IN THE VPST        */VP 00110
                     80 LEVEL FIXED BIN,          /* LEVEL                    */VP 00120
                     80 VTIME FIXED BIN (31)                                      VP 00130
                  ;                                                             SHE00080
400090   1  0  DCL (EPT,SPT) PTR;                                                SHE00090
400100   1  0  DCL 1 EVENT BASED (EPT),                                          SHE00100
                  70 NEXT PTR,                 /* NEXT EVENT               */ EVE00010
                  70 STIME FIXED BIN (31),                                      EVE00020
                  70 TYPE CHAR (12) VAR,                                        EVE00030
                  70 INDEX FIXED BIN (31),                                      EVE00040
                  70 PTR PTR                                                    EVE00050
                  ;                                                             SHE00100
500110   1  0  DCL 1 SVR BASED (SPT),                                           SHE00110
                  70 NEXT PTR,                                                  SVR00010
                  70 STIME FIXED BIN (31),                                      SVR00020
                  70 STIMEQ FIXED BIN (31),    /* ACCUMULATED STIME        */SVR00030
                  70 RTIME FIXED BIN (31),                                      SVR00040
                  70 RRATE FIXED BIN (15,7),   /* REAL CPU TICK RATE       */SVR00050
                  70 ID FIXED BIN                                               SVR00060
```

```
NUMBER LEV NT                                                                    R

                    ;                                                            SHE00110
600120   1  0   DCL (I,LEVEL,EARLIEST) FIXED BIN;                                SHE00120
600130   1  0   DCL (STM,STM0) FIXED BIN (31);                                   SHE00130
600140   1  0   DCL DAT CHAR(6);                                                 SHE00140
600150   1  0   DCL TIM CHAR(9);                                                 SHE00150
600160   1  0   DCL (YR,MO,DY,HR,MN) CHAR(2);                                    SHE00160
600170   1  0   DCL STATS FILE OUTPUT EXTERNAL;                                  SHE00170
700010   1  0   DCL 1 DEBUG EXTERNAL STATIC,                                     DEB00010
                    2 SLEEP$ BIT(1) INIT ('0'B),                                 DEB00020
                    2 SAHER$ BIT(1) INIT ('0'B),                                 DEB00030
                    2 SCHEDULER$ BIT(1) INIT ('0'B),                             DEB00040
                    2 SHELL$ BIT(1) INIT ('0'B),                                 DEB00050
                    2 AAHER$ BIT(1) INIT ('0'B);                                 DEB00060
                                                                                 SHE00190
700200   1  0   DCL (ADDR,NULL,MAX,MOD,DATE,TIME,SUBSTR) BUILTIN;               SHE00200
700210   1  0   DCL (ADDQ,DEBUGR) ENTRY;                                        SHE00210
800010   1  0   DCL RTIMER ENTRY RETURNS (FIXED BIN (31));                      RTI00010
900010   1  0   DCL QEVENT ENTRY (FIXED BIN, FIXED BIN (31), CHAR(*) VAR,      QEV00010
                          FIXED BIN (31), PTR);                                  QEV00020
900240   1  0   DCL (GBER,GCER,SAHER,AAHER) ENTRY;                              SHE00240
                                                                                 SHE00250
                                                                                 SHE00260
900270   1  0       OPEN FILE (STATS) TITLE ('STATS');                          SHE00270
900280   1  0       DAT = DATE;                                                 SHE00280
900290   1  0       TIM = TIME;                                                 SHE00290
900300   1  0       YR = SUBSTR(DAT,1,2);                                       SHE00300
900310   1  0       MO = SUBSTR(DAT,3,2);                                       SHE00310
900320   1  0       DY = SUBSTR(DAT,5,2);                                       SHE00320
900330   1  0       HR = SUBSTR(TIM,1,2);                                       SHE00330
900340   1  0       MN = SUBSTR(TIM,3,2);                                       SHE00340
900350   1  0       PUT FILE (STATS) SKIP EDIT (' *** ',YR,'/',MO,'/',DY,' ',   SHE00350
                      HR,':',MN)                                                 SHE00360
                      (A,A(2),A,A(2),A,A(2),A,A(2),A,A(2));                      SHE00370
900380   1  0       PUT FILE (STATS) SKIP EDIT (' *** PARAMETERS: ') (A);       SHE00380
900390   1  0       PUT FILE (STATS) SKIP (2);                                  SHE00390
900400   1  0       CALL DEBUGR;                                                SHE00400
900410   1  0       CALL SET_LOS;                                               SHE00410
900420   1  0       CALL SET_Q;                                                 SHE00420
900430   1  0       CALL SET_P;                                                 SHE00430
900440   1  0       PUT SKIP (3) LIST (' START SIMULATION...');                 SHE00440
900450   1  0       PUT SKIP (3);                                               SHE00450
900460   1  0       PUT FILE (STATS) SKIP (3) EDIT (' *** SIMULATION:') (A);    SHE00460
900470   1  0       PUT FILE (STATS) SKIP (2);                                  SHE00470
                                                                                 SHE00480
900490   1  0   LOOP:                                                           SHE00490
                                                                                 SHE00500
                    EARLIEST = -1;                                               SHE00510
900520   1  0       STM0 = 2147483647;                                          SHE00520
                                                                                 SHE00530
```

```
NUMBER LEV NT

900540   1  0       DO I = 0 TO        6;
900550   1  1          IF (EQUEUE(I) ^= NULL) THEN DO;
900560   1  2             EPT = EQUEUE(I);
900570   1  2             SPT = SQUEUE(I);
900580   1  2             STM = MAX (EVENT.STIME,SVR.STIME);
900590   1  2             IF (STM < STMO) THEN DO;
900600   1  3                EARLIEST = I;
900610   1  3                STMO = STM;
900620   1  3             END;
900630   1  2          END;
900640   1  1       END;

900660   1  0       IF (DEBUG.SHELL$) THEN DO;
900670   1  1          PUT SKIP EDIT (' SHELL: EARLIEST = ',EARLIEST) (A,F(2));
900680   1  1       END;
900690   1  0       IF (EARLIEST = -1) THEN GOTO EXIT;

900710   1  0       EPT = EQUEUE(EARLIEST);
900720   1  0       SPT = SQUEUE(EARLIEST);
900730   1  0       EQUEUE(EARLIEST) = EVENT.NEXT;
900740   1  0       LEVEL = (EARLIEST-1)/3;
900750   1  0       SVR.STIME = STMO;
900760   1  0       SVR.RTIME = RTIMER;

900780   1  0       IF (EARLIEST = 0) THEN DO;
900790   1  1          CALL GBER (SVR,EVENT);
900800   1  1       END;
900810   1  0       ELSE IF (MOD(EARLIEST,3) = 1) THEN DO; /* GC */
900820   1  1          CALL GCER (LOSES(LEVEL),SVR,EVENT);
900830   1  1       END;
900840   1  0       ELSE IF (MOD(EARLIEST,3) = 2) THEN DO; /* AAH */
900850   1  1          CALL AAHER (LOSES(LEVEL),SVR,EVENT);
900860   1  1       END;
900870   1  0       ELSE DO; /* SAH */
900880   1  1          CALL SAHER (LOSES(LEVEL),SVR,EVENT);
900890   1  1       END;

900910   1  0       FREE EVENT;
900920   1  0       IF (EARLIEST ^= 0) & (MOD(EARLIEST,3) ^= 1) THEN DO;
900930   1  1          SQUEUE(EARLIEST) = SVR.NEXT;
900940   1  1          SVR.NEXT = NULL;
900950   1  1          CALL ADDQ (SQUEUE(EARLIEST),SPT);
900960   1  1       END;

900980   1  0       GOTO LOOP;

901000   1  0  EXIT:
                    PUT FILE (STATS) SKIP (3) EDIT (' *** STATISTICS:') (A);
```

PL/I OPTIMIZING COMPILER SHELL: PROC OPTIONS (MAIN);

```
NUMBER LEV NT                                                                    R

901030   1  0      PUT FILE (STATS) SKIP (2);                                    SHE01030
901040   1  0      CALL REPORT;                                                  SHE01040
901050   1  0      CLOSE FILE (STATS);                                           SHE01050
901060   1  0      RETURN;                                                       SHE01060
                                                                                SHE01070
                                                                                SHE01080
901090   1  0  SET_LOS: PROC;                                                   SHE01090
                                                                                SHE01100
901110   2  0  DCL (L,IV) FIXED BIN;                                           SHE01110
                                                                                SHE01120
901130   2  0      DO L = 0 TO        1;                                        SHE01130  1
901140   2  1         LOSES(L).LEVEL = L;                                       SHE01140
901150   2  1         LOSES(L).SRQ.SIZE = 0;                                    SHE01150
901160   2  1         LOSES(L).SRQ.MAX = 0;                                     SHE01160
901170   2  1         LOSES(L).OBB.SIZE = 0;                                    SHE01170
901180   2  1         LOSES(L).OBB.MAX = 0;                                     SHE01180
901190   2  1         DO IV = 1 TO       20;                                    SHE01190  1
901200   2  2            LOSES(L).VPS.TABLE(IV).LEVEL = L;                      SHE01200
901210   2  2            LOSES(L).VPS.TABLE(IV).VPID = IV;                      SHE01210
901220   2  2            LOSES(L).VPS.TABLE(IV).STATUS = 'VOID';                SHE01220
901230   2  2         END;                                                      SHE01230
901240   2  1      END;                                                         SHE01240
                                                                                SHE01250
901260   2  0  END SET_LOS;                                                    SHE01260
                                                                                SHE01270
                                                                                SHE01280
901290   1  0  SET_Q: PROC;                                                    SHE01290
                                                                                SHE01300
901310   2  0  DCL (IQ,L,NP,IP) FIXED BIN;                                    SHE01310
901320   2  0  DCL (RRATE) FIXED BIN (15,7);                                  SHE01320
                                                                                SHE01330
901340   2  0      DO IQ = 0 TO       6;                                        SHE01340  1
901350   2  1         EQUEUE(IQ) = NULL;                                        SHE01350
901360   2  1         SQUEUE(IQ) = NULL;                                        SHE01360
901370   2  1      END;                                                         SHE01370
                                                                                SHE01380
901390   2  0      DO L = 0 TO        1;                                        SHE01390  1
901400   2  1         PUT SKIP EDIT (' LEVEL ',L,': ') (A,F(2),A);              SHE01400
901410   2  1         PUT FILE (STATS) SKIP EDIT (' LEVEL ',L,': ') (A,F(2),A); SHE01410
901420   2  1         PUT SKIP LIST ('  NO. OF PROCESSORS, RRATE> ');           SHE01420
901430   2  1         GET LIST (NP,RRATE);                                      SHE01430
901440   2  1         PUT FILE (STATS) DATA (NP,RRATE);                         SHE01440
901450   2  1         DO IP = 1 TO NP;                                          SHE01450
901460   2  2            CALL NEW_SVR(RRATE,IP,SQUEUE(L*3+2));                  SHE01460
901470   2  2            CALL NEW_SVR(RRATE,IP,SQUEUE(L*3+3));                  SHE01470
901480   2  2         END;                                                      SHE01480
901490   2  1         CALL NEW_SVR(RRATE,1,SQUEUE(L*3+1));                      SHE01490
901500   2  1      END; /* L */                                                 SHE01500
                                                                                SHE01510
```

```
NUMBER LEV NT                                                                    R

901520   2  0       CALL NEW_SVR(1,1,SQUEUE(0));                                 SHE01520
901530   2  0       CALL QEVENT (3,0,'INIT',0,NULL);                             SHE01530
901540   2  0       RETURN;                                                      SHE01540
                    /* END */                                                   SHE01550
                                                                                SHE01560
                                                                                SHE01570
901580   2  0  NEW_SVR: PROC (RRATE,ID,WHERE);                                  SHE01580
                                                                                SHE01590
                                                                                SHE01600
901600   3  0  DCL RRATE FIXED BIN (15,7);                                      SHE01610
901610   3  0  DCL ID FIXED BIN;                                                SHE01620
901620   3  0  DCL WHERE PTR;                                                   SHE01630
901630   3  0  DCL SVRPT PTR;                                                  SHE01640
901640   3  0  DCL 1 SVR BASED (SVRPT),                                         SVR00010
                  70 NEXT PTR,                                                  SVR00020
                  70 STIME FIXED BIN (31),                                      SVR00030
                  70 STIMEQ FIXED BIN (31),       /* ACCUMULATED STIME */       SVR00040
                  70 RTIME FIXED BIN (31),                                      SVR00050
                  70 RRATE FIXED BIN (15,7),      /* REAL CPU TICK RATE */      SVR00060
                  70 ID FIXED BIN                                               SHE01640
                ;                                                               SHE01650
                                                                                SHE01660
1001660  3  0       ALLOCATE SVR;                                              SHE01670
1001670  3  0       SVR.STIME = 0;                                             SHE01680
1001680  3  0       SVR.STIMEQ = 0;                                            SHE01690
1001690  3  0       SVR.RRATE = RRATE;                                         SHE01700
1001700  3  0       SVR.ID = ID;                                               SHE01710
1001710  3  0       SVR.NEXT = WHERE;                                          SHE01720
1001720  3  0       WHERE = SVRPT;                                             SHE01730
                                                                                SHE01740
1001740  3  0  END NEW_SVR;                                                     SHE01750
                                                                                SHE01760
1001770  2  0  END SET_Q;                                                       SHE01770
                                                                                SHE01780
                                                                                SHE01790
1001800  1  0  SET_P: PROC;                                                     SHE01800
                                                                                SHE01810
1100010  2  0  DCL 1 PARAMS EXTERNAL STATIC,                                    PAR00010
                  2 THRU_RATE FIXED BIN (31,7),                                 PAR00020
                  2 DELAY_GB_GC FIXED BIN (31),                                 PAR00030
                  2 DELAY_GC_GB FIXED BIN (31),                                 PAR00040
                  2 TERMINALS FIXED BIN;                                        PAR00050
                                                                                SHE01830
1101840  2  0       PUT SKIP LIST (' THRU_RATE,GB_GC,GC_GB,TERMINALS> ');      SHE01840
1101850  2  0       GET LIST (PARAMS.THRU_RATE,PARAMS.DELAY_GB_GC,             SHE01850
                           PARAMS.DELAY_GC_GB,PARAMS.TERMINALS);                SHE01860
1101870  2  0       PUT FILE (STATS) SKIP EDIT (' ') (A);                      SHE01870
1101880  2  0       PUT FILE (STATS) DATA (PARAMS);                            SHE01880
                                                                                SHE01890
```

```
PL/I OPTIMIZING COMPILER        SHELL: PROC OPTIONS (MAIN);

                                                                          R
 NUMBER LEV NT
                                                                   SHE01900
 1101900   2  0  END SET_P;                                        SHE01910
                                                                   SHE01920
                                                                   SHE01930
 1101930   1  0  REPORT: PROC;                                     SHE01940
                                                                   SHE01950
 1101950   2  0  DCL SVRPT PTR;                                    SHE01960
 1101960   2  0  DCL 1 SVR BASED (SVRPT),                          SVR00010
                       70 NEXT PTR,                                SVR00020
                       70 STIME FIXED BIN (31),                    SVR00030
                       70 STIMEQ FIXED BIN (31),       /* ACCUMULATED STIME      */SVR00030
                       70 RTIME FIXED BIN (31),                    SVR00040
                       70 RRATE FIXED BIN (15,7),      /* REAL CPU TICK RATE     */SVR00050
                       70 ID FIXED BIN                             SVR00060
                 ;                                                 SHE01960
 1201970   2  0  DCL (I,L) FIXED BIN;                              SHE01970
 1201980   2  0  DCL TITLE(3) CHAR(3) STATIC INIT (' GC','AAH','SAH');  SHE01980
                                                                   SHE01990
 1202000   2  0      DO L = 0 TO        1;                         SHE02000    1
                                                                   SHE02010
 1202020   2  1         PUT FILE (STATS) SKIP EDIT (' LEVEL ',L,': ') (A,F(2),A);  SHE02020
 1202030   2  1         PUT FILE (STATS) SKIP EDIT ('   ',         SHE02030
                           ' SRQ.MAX =',LOSES(L).SRQ.MAX,          SHE02040
                           ' DBB.MAX =',LOSES(L).DBB.MAX)          SHE02050
                           (A,A,F(11),A,F(11));                    SHE02060
                                                                   SHE02070
                                                                   SHE02080
 1202080   2  1         DO I = 1 TO 3;                             SHE02090
 1202090   2  2            PUT FILE (STATS) SKIP EDIT ('  ',TITLE(I),':') (A,A(3),A);  SHE02100
 1202100   2  2            SVRPT = SQUEUE(L*3+I);                  SHE02110
 1202110   2  2            DO WHILE (SVRPT ^= NULL);               SHE02120
 1202120   2  3               PUT FILE (STATS) SKIP EDIT ('   ') (A);  SHE02130
 1202130   2  3               PUT FILE (STATS) DATA (SVR.ID,SVR.STIME,SVR.STIMEQ);  SHE02140
 1202140   2  3               SVRPT = SVR.NEXT;                    SHE02150
 1202150   2  3            END;                                    SHE02160
 1202160   2  2         END;                                       SHE02170
                                                                   SHE02180
 1202180   2  1      END; /* L */                                  SHE02190
                                                                   SHE02200
 1202200   2  0  END REPORT;                                       SHE02210
                                                                   SHE02220
 1202220   1  0  END SHELL;
```

SOURCE LISTING

```
NUMBER LEV NT                                                                    R

    10       0  SLEEP: PROC;                                                     SLE00010
                                                                                 SLE00020
    30   1   0  DCL (THIS,NEXT) PTR;                                            SLE00030
    40   1   0  DCL (NULL,ADDR,MIN) BUILTIN;                                    SLE00040
    50   1   0  DCL (NSIZE,OSIZE,NEW_NBLK,OLD_NBLK,DIFF_NBLK) FIXED BIN (31);  SLE00050
    60   1   0  DCL SAVE_ADDR FIXED BIN (31);                                   SLE00060
100010   1   0  DCL THISVP PTR EXTERNAL STATIC;        /* -> CURRENT VP, SET BY VPER */VPX00010
                                                                                 VPX00020
100030   1   0  DCL 1 VP BASED (THISVP),                                        VPX00030
                    80 BDSA FIXED BIN (31),       /* ADDR (BOTTOM DSA)          */VP 00010
                    80 TDSA FIXED BIN (31),       /* ADDR (TOP DSA)             */VP 00020
                    80 SAVESIZE FIXED BIN (31),   /* SIZE OF AREA TO BE SAVED   */VP 00030
                    80 SAFE PTR,                  /* -> 1ST SAVBLK              */VP 00040
                    80 PROCNAME CHAR (7) VAR,     /* NAME OF TOP LEVEL PROC     */VP 00050
                    80 STATUS CHAR (12) VAR,                                      VP 00060
                    80 MAIL PTR,                  /* CHAIN OF INCOMING MAIL     */VP 00070
                    80 WAIT,                                                      VP 00080
                       81 BOX FIXED BIN,          /* BOX AWAITING MAIL          */VP 00090
                       81 MSG PTR,                /* MSG IN WAIT.BOX            */VP 00100
                    80 VPID FIXED BIN,            /* INDEX IN THE VPST          */VP 00110
                    80 LEVEL FIXED BIN,           /* LEVEL                      */VP 00120
                    80 VTIME FIXED BIN (31)                                       VP 00130
                ;                                                                VPX00030
                                                                                 SAV00030
                                                                                 SAV00060
400070   1   0  DCL NEWSAVBLK PTR;                                               SAV00070
                                                                                 SAV00080
400090   1   0  DCL 1 SAVBLK BASED (NEWSAVBLK),                                  SAV00090
                    2 NEXT PTR,                   /* NEXT SAVED ITEM            */SAV00100
                    2 ADDR FIXED BIN (31),        /* BEG ADDR OF SAVED ITEM     */SAV00110
                    2 SIZE FIXED BIN (31),        /* SIZE OF SAVED ITEM         */SAV00120
                    2 AREA (     250) FIXED BIN (31);                            SAV00130  1
                                                                                 SAV00140
500010   1   0  DCL 1 DEBUG EXTERNAL STATIC,                                     DEB00010
                    2 SLEEP$ BIT(1) INIT ('0'B),                                 DEB00020
                    2 SAHER$ BIT(1) INIT ('0'B),                                 DEB00030
                    2 SCHEDULER$ BIT(1) INIT ('0'B),                             DEB00040
                    2 SHELL$ BIT(1) INIT ('0'B),                                 DEB00050
                    2 AAHER$ BIT(1) INIT ('0'B);                                 DEB00060
                                                                                 SLE00100
600010   1   0  DCL DSA FIXED BIN (31,0),                                       GET00010
                   NAB FIXED BIN (31,0),                                         GET00020
                   SEG FIXED BIN (31,0),                                         GET00030
                   EOS FIXED BIN (31,0);                                         GET00040
                                                                                 GET00050
                                                                                 GET00060
```

```
NUMBER LEV NT                                                                    R

600070   1  0  DCL GET4 ENTRY (FIXED BIN (31), FIXED BIN (31),                    GET00070
                            FIXED BIN (31), FIXED BIN (31))                     GET00080
                     OPTIONS (ASSEMBLER, INTER);                                GET00090
700010   1  0  DCL SVSTK ENTRY (PTR, FIXED BIN (31))                            SVS00010
                      OPTIONS (ASSEMBLER, INTER);                               SVS00020
                                                                                SVS00030
                    /* ARG1: PTR -> SAVBLK                                     SVS00040
                       ARG2: ADDR OF RETURN DSA (BOTTOM DSA)                    SVS00050
                    */                                                          SVS00060
                                                                                SLE00130
                   /* VP.STATUS = 'RUNNABLE' DEBUG */                           SLE00140
                                                                                SLE00150
700160   1  0      CALL GET4 (DSA,NAB,SEG,EOS);                                 SLE00160
700170   1  0      IF SEG ^= 255 THEN DO;                                       SLE00170
700180   1  1         PUT SKIP LIST (' ERROR: ILLEGAL SEGNO. (SLEEP)');         SLE00180
700190   1  1         PUT DATA (SEG);                                           SLE00190
700200   1  1         STOP;                                                     SLE00200
700210   1  1      END;                                                         SLE00210
                                                                                SLE00220
                                                                                SLE00230
700240   1  0      NSIZE = NAB - VP.BDSA;                                       SLE00240
700250   1  0      OSIZE = VP.SAVESIZE;                                         SLE00250
700260   1  0      NEW_NBLK = (NSIZE +     1000 - 1) /     1000;                SLE00260  1
700270   1  0      OLD_NBLK = (OSIZE +     1000 - 1) /     1000;                SLE00270  1
700280   1  0      DIFF_NBLK = NEW_NBLK - OLD_NBLK;                             SLE00280
700290   1  0      IF (DEBUG.SLEEP$) THEN DO;                                   SLE00290
700300   1  1         PUT SKIP LIST (' SLEEP: ');                               SLE00300
700310   1  1         PUT DATA (NSIZE,OSIZE,NEW_NBLK,OLD_NBLK,DIFF_NBLK);       SLE00310
700320   1  1      END;                                                         SLE00320
                                                                                SLE00330
700340   1  0      IF DIFF_NBLK > 0 THEN DO;                                    SLE00340
700350   1  1         NEXT = VP.SAFE;                                           SLE00350
700360   1  1         DO WHILE (DIFF_NBLK > 0);                                 SLE00360
700370   1  2            ALLOCATE SAVBLK SET (THIS);                            SLE00370
700380   1  2            THIS -> SAVBLK.NEXT = NEXT;                            SLE00380
700390   1  2            NEXT = THIS;                                           SLE00390
700400   1  2            DIFF_NBLK = DIFF_NBLK - 1;                             SLE00400
700410   1  2         END;                                                      SLE00410
700420   1  1         VP.SAFE = THIS;                                           SLE00420
700430   1  1      END;                                                         SLE00430
700440   1  0      ELSE IF DIFF_NBLK < 0 THEN DO;                               SLE00440
700450   1  1         THIS = VP.SAFE;                                           SLE00450
700460   1  1         DO WHILE (DIFF_NBLK < 0);                                 SLE00460
700470   1  2            NEXT = THIS -> SAVBLK.NEXT;                            SLE00470
700480   1  2            FREE THIS -> SAVBLK;                                   SLE00480
700490   1  2            THIS = NEXT;                                           SLE00490
700500   1  2            DIFF_NBLK = DIFF_NBLK + 1;                             SLE00500
700510   1  2         END;                                                      SLE00510
700520   1  1         VP.SAFE = THIS;                                           SLE00520
```

```
NUMBER LEV NT                                                                    R

700530   1  1     END;                                                          SLE00530
                                                                               SLE00540
700550   1  0     VP.SAVESIZE = NSIZE;                                          SLE00550
700560   1  0     VP.TDSA = DSA;                                                SLE00560
                                                                               SLE00570
700580   1  0     THIS = VP.SAFE;                                               SLE00580
700590   1  0     SAVE_ADDR = VP.BDSA;                                          SLE00590
700600   1  0     DO WHILE (THIS ^= NULL);                                      SLE00600
700610   1  1        THIS -> SAVBLK.SIZE = MIN(    1000,NSIZE);               SLE00610  1
700620   1  1        THIS -> SAVBLK.ADDR = SAVE_ADDR;                           SLE00620
700630   1  1        NSIZE = NSIZE -     1000;                                  SLE00630  1
700640   1  1        SAVE_ADDR = SAVE_ADDR +     1000;                          SLE00640  1
700650   1  1        THIS = THIS -> SAVBLK.NEXT;                                SLE00650
700660   1  1     END;                                                          SLE00660
                                                                               SLE00670
700680   1  0     CALL SVSTK (VP.SAFE,VP.BDSA);                                 SLE00680
                                                                               SLE00690
                  /* CONTROL SHOULD COME HERE AFTER RTSTK */                   SLE00700
                                                                               SLE00710
700720   1  0  END SLEEP;                                                       SLE00720
```

SOURCE LISTING

```
NUMBER LEV NT

    10      0  STIMER: PROC RETURNS (FIXED BIN(31));                                  STI00010
                                                                                      STI00020
               %INCLUDE SVRX;*****************************************************STI00030
100010   1  0  DCL THISSVR PTR EXTERNAL STATIC;   /* -> CURRENT SVR, SET BY SAHER */  SVR00010
                                                                                      SVR00020
100030   1  0  DCL 1 SVR BASED (THISSVR), %INCLUDE SVR;*****************************SVR00030
                    70 NEXT PTR,                                                      SVR00010
                    70 STIME FIXED BIN (31),                                          SVR00020
                    70 STIMEQ FIXED BIN (31),        /* ACCUMULATED STIME          */SVR00030
                    70 RTIME FIXED BIN (31),                                          SVR00040
                    70 RRATE FIXED BIN (15,7),       /* REAL CPU TICK RATE         */SVR00050
                    70 ID FIXED BIN                                                   SVR00060
               ***************************************   ;                          SVR00030
                                                                                      SVR00040
               **************                                                         STI00030
400040   1  0  DCL RT FIXED BIN (31);                                                STI00040
               %INCLUDE RTIMER;***************************************************STI00050
500010   1  0  DCL RTIMER ENTRY RETURNS (FIXED BIN (31));                           RTI00010
               ****************                                                       STI00050
                                                                                      STI00060
500070   1  0      RT = RTIMER - SVR.RTIME;                                          STI00070
500080   1  0      RT = RT * SVR.RRATE;                                              STI00080
500090   1  0      RETURN (RT + SVR.STIME);                                          STI00090
                                                                                     STI00100
500110   1  0  END;                                                                  STI00110
```

SOURCE LISTING

```
NUMBER LEV NT                                                                    R

    10       0  SVCER: PROC (SVC,PT);                                          SVC00010
                                                                               SVC00020
    30   1   0  DCL SVC FIXED BIN;                                            SVC00030
    40   1   0  DCL PT PTR;                                                   SVC00040
                                                                               SVC00050
100010   1   0  DCL THISVP PTR EXTERNAL STATIC;        /* -> CURRENT VP, SET BY VPER */VPX00010
                                                                               VPX00020
100030   1   0  DCL 1 VP BASED (THISVP),                                       VPX00030
                     80 BDSA FIXED BIN (31),       /* ADDR (BOTTOM DSA)        */VP 00010
                     80 TDSA FIXED BIN (31),       /* ADDR (TOP DSA)           */VP 00020
                     80 SAVESIZE FIXED BIN (31),   /* SIZE OF AREA TO BE SAVED */VP 00030
                     80 SAFE PTR,                  /* -> 1ST SAVBLK            */VP 00040
                     80 PROCNAME CHAR (7) VAR,     /* NAME OF TOP LEVEL PROC   */VP 00050
                     80 STATUS CHAR (12) VAR,                                    VP 00060
                     80 MAIL PTR,                  /* CHAIN OF INCOMING MAIL   */VP 00070
                     80 WAIT,                                                    VP 00080
                        81 BOX FIXED BIN,          /* BOX AWAITING MAIL        */VP 00090
                        81 MSG PTR,                /* MSG IN WAIT.BOX          */VP 00100
                     80 VPID FIXED BIN,            /* INDEX IN THE VPST        */VP 00110
                     80 LEVEL FIXED BIN,           /* LEVEL                    */VP 00120
                     80 VTIME FIXED BIN (31)                                     VP 00130
                   ;                                                           VPX00030
400010   1   0  DCL THISLOS PTR EXTERNAL STATIC;    /* -> CURRENT LOS, ST BY SAHER */LOS00010
                                                                               LOS00020
400030   1   0  DCL 1 LOS BASED (THISLOS),                                      LOS00030
                   70 LEVEL FIXED BIN,                                         LOS00010
                   70 SRQ,                                                     LOS00020
                      71 SIZE FIXED BIN,                                       LOS00030
                      71 MAX FIXED BIN,                                        LOS00040
                   70 DBB,                                                     LOS00050
                      71 SIZE FIXED BIN,                                       LOS00060
                      71 MAX FIXED BIN,                                        LOS00070
                   70 VPS,                                                     LOS00080
                      71 TABLE (      20),                                     LOS00090  1
                      80 BDSA FIXED BIN (31),    /* ADDR (BOTTOM DSA)        */VP 00010
                      80 TDSA FIXED BIN (31),    /* ADDR (TOP DSA)           */VP 00020
                      80 SAVESIZE FIXED BIN (31),/* SIZE OF AREA TO BE SAVED */VP 00030
                      80 SAFE PTR,               /* -> 1ST SAVBLK            */VP 00040
                      80 PROCNAME CHAR (7) VAR,  /* NAME OF TOP LEVEL PROC   */VP 00050
                      80 STATUS CHAR (12) VAR,                                 VP 00060
                      80 MAIL PTR,               /* CHAIN OF INCOMING MAIL   */VP 00070
                      80 WAIT,                                                 VP 00080
                         81 BOX FIXED BIN,       /* BOX AWAITING MAIL        */VP 00090
                         81 MSG PTR,             /* MSG IN WAIT.BOX          */VP 00100
                      80 VPID FIXED BIN,         /* INDEX IN THE VPST        */VP 00110
                      80 LEVEL FIXED BIN,        /* LEVEL                    */VP 00120
```

```
 NUMBER LEV NT                                                                                    R

                        80 VTIME FIXED BIN (31)                                                  VP 00130
                      ;                                                                          LOS00030
 800010   1  0  DCL THISSVR PTR EXTERNAL STATIC;   /* -> CURRENT SVR, SET BY SAHER */           SVR00010
                                                                                                 SVR00020
 800030   1  0  DCL 1 SVR BASED (THISSVR),                                                       SVR00030
                   70 NEXT PTR,                                                                  SVR00010
                   70 STIME FIXED BIN (31),                                                      SVR00020
                   70 STIMEQ FIXED BIN (31),        /* ACCUMULATED $TIME                     */SVR00030
                   70 RTIME FIXED BIN (31),                                                      SVR00040
                   70 RRATE FIXED BIN (15,7),       /* REAL CPU TICK RATE                    */SVR00050
                   70 ID FIXED BIN                                                               SVR00060
                  ;                                                                              SVR00030
                                                                                                 SVC00090
 1000100  1  0  DCL SVCS (1) LABEL INIT (START);                                                 SVC00100
                                                                                                 SVC00110
 1000120  1  0  DCL IV FIXED BIN;                                                               SVC00120
 1100010  1  0  DCL PT_SVC PTR;                                                                 PFS00010
                                                                                                 PFS00020
 1100030  1  0  DCL 1 PF_SVC BASED (PT_SVC),                                                    PFS00030
                   2 SVC CHAR (7) VAR,                                                           PFS00040
                   2 PTR PTR;                                                                    PFS00050
 1200010  1  0  DCL VPSTART ENTRY (/* VP */, CHAR(*) VAR, PTR);                                 VPS00010
                                                                                                 SVC00150
 1200160  1  0      IF (SVC <= 0) | (SVC > 1) THEN DO;                                          SVC00160
 1200170  1  1        PUT SKIP LIST (' ERROR: ILLEGAL SVC.');                                   SVC00170
 1200180  1  1        PUT DATA (SVC);                                                           SVC00180
 1200190  1  1        STOP;                                                                     SVC00190
 1200200  1  1      END;                                                                        SVC00200
                                                                                                 SVC00210
 1200220  1  0      GOTO SVCS(SVC);                                                             SVC00220
                                                                                                 SVC00230
 1200240  1  0  START: /* SVC = 1 */                                                            SVC00240
                                                                                                 SVC00250
                     DO IV = 1 TO       20;                                                      SVC00260
 1200270  1  1        IF LOS.VPS.TABLE(IV).STATUS = 'VOID' THEN GOTO START1;                    SVC00270
 1200280  1  1      END;                                                                        SVC00280
 1200290  1  0      PUT SKIP LIST (' ERROR: EXCEED MAX. VP. (SVCER)');                          SVC00290
 1200300  1  0      STOP;                                                                       SVC00300
                                                                                                 SVC00310
 1200320  1  0  START1:                                                                         SVC00320
                                                                                                 SVC00330
                     PT_SVC = PT;                                                                SVC00340
 1200350  1  0      CALL VPSTART (LOS.VPS.TABLE(IV), PF_SVC.SVC, PF_SVC.PTR);                   SVC00350
 1200360  1  0      FREE PF_SVC;                                                                SVC00360
 1200370  1  0      RETURN;                                                                     SVC00370
                                                                                                 SVC00380
 1200390  1  0  END;                                                                            SVC00390
```

SOURCE LISTING

```
NUMBER LEV NT

    10      0  SYNC: PROC;                                                          SYN00010
                                                                                    SYN00020
               %INCLUDE VPX;*************************************************SYN00030
100010   1   0  DCL THISVP PTR EXTERNAL STATIC;        /* -> CURRENT VP, SET BY VPER */VPX00010
                                                                                    VPX00020
100030   1   0  DCL 1 VP BASED (THISVP), %INCLUDE VP;*****************************VPX00030
                      80 BDSA FIXED BIN (31),       /* ADDR (BOTTOM DSA)          */VP 00010
                      80 TDSA FIXED BIN (31),       /* ADDR (TOP DSA)             */VP 00020
                      80 SAVESIZE FIXED BIN (31),   /* SIZE OF AREA TO BE SAVED   */VP 00030
                      80 SAFE PTR,                  /* -> 1ST SAVBLK              */VP 00040
                      80 PROCNAME CHAR (7) VAR,     /* NAME OF TOP LEVEL PROC     */VP 00050
                      80 STATUS CHAR (12) VAR,                                      VP 00060
                      80 MAIL PTR,                  /* CHAIN OF INCOMING MAIL     */VP 00070
                      80 WAIT,                                                      VP 00080
                         81 BOX FIXED BIN,          /* BOX AWAITING MAIL          */VP 00090
                         81 MSG PTR,                /* MSG IN WAIT.BOX            */VP 00100
                      80 VPID FIXED BIN,            /* INDEX IN THE VPST          */VP 00110
                      80 LEVEL FIXED BIN,           /* LEVEL                      */VP 00120
                      80 VTIME FIXED BIN (31)                                       VP 00130
               ************************************    ;                          VPX00030
                                                                                    VPX00040
               ************                                                         SYN00030
                                                                                    SYN00040
               %INCLUDE QEVENT;*************************************************SYN00050
500010   1   0  DCL QEVENT ENTRY (FIXED BIN, FIXED BIN (31), CHAR(*) VAR,          QEV00010
                                 FIXED BIN (31), PTR);                              QEV00020
               ***************                                                      SYN00050
500060   1   0  DCL NULL BUILTIN;                                                  SYN00060
500070   1   0  DCL SLEEP ENTRY;                                                   SYN00070
               %INCLUDE STIMER;*************************************************SYN00080
600010   1   0  DCL STIMER ENTRY RETURNS (FIXED BIN (31));                        STI00010
               ***************                                                      SYN00080
                                                                                    SYN00090
600100   1   0      CALL QEVENT (VP.LEVEL*3+3,STIMER,'SYNC',VP.VPID,NULL);        SYN00100
600110   1   0      CALL SLEEP;                                                   SYN00110
                                                                                    SYN00120
600130   1   0  END;                                                              SYN00130
```

SOURCE LISTING

```
NUMBER LEV NT

    10      0  TERM: PROC;                                                          TER00010
                                                                                    TER00020
               %INCLUDE USERS;*************************************************TER00030
               %INCLUDE STIMER;************************************************USE00010
200010    1  0  DCL STIMER ENTRY RETURNS (FIXED BIN (31));                         STI00010
               ***************                                                      USE00010
300020    1  0  DCL SEND ENTRY (FIXED BIN, FIXED BIN, FIXED BIN, CHAR(*) VAR,     USE00020
                        FIXED BIN, PTR);                                          USE00030
300040    1  0  DCL WAIT ENTRY (FIXED BIN);                                        USE00040
300050    1  0  DCL SYNC ENTRY;                                                    USE00050
300060    1  0  DCL FINISH ENTRY;                                                  USE00060
                                                                                  USE00070
               ***************                                                    TER00030
400040    1  0  DCL (NULL,LENGTH) BUILTIN;                                        TER00040
                                                                                  TER00050
               %INCLUDE VPX;***************************************************TER00060
500010    1  0  DCL THISVP PTR EXTERNAL STATIC;        /* -> CURRENT VP, SET BY VPER */VPX00010
                                                                                  VPX00020
500030    1  0  DCL 1 VP BASED (THISVP), %INCLUDE VP;***************************VPX00030
                    80 BDSA FIXED BIN (31),       /* ADDR (BOTTOM DSA)          */VP 00010
                    80 TDSA FIXED BIN (31),       /* ADDR (TOP DSA)             */VP 00020
                    80 SAVESIZE FIXED BIN (31),   /* SIZE OF AREA TO BE SAVED   */VP 00030
                    80 SAFE PTR,                  /* -> 1ST SAVBLK              */VP 00040
                    80 PROCNAME CHAR (7) VAR,     /* NAME OF TOP LEVEL PROC     */VP 00050
                    80 STATUS CHAR (12) VAR,                                      VP 00060
                    80 MAIL PTR,                  /* CHAIN OF INCOMING MAIL     */VP 00070
                    80 WAIT,                                                      VP 00080
                       81 BOX FIXED BIN,          /* BOX AWAITING MAIL          */VP 00090
                       81 MSG PTR,                /* MSG IN WAIT.BOX            */VP 00100
                    80 VPID FIXED BIN,            /* INDEX IN THE VPST          */VP 00110
                    80 LEVEL FIXED BIN,           /* LEVEL                      */VP 00120
                    80 VTIME FIXED BIN (31)                                       VP 00130
               ************************************    ;                          VPX00030
                                                                                  VPX00040
               *************                                                      TER00060
               %INCLUDE SVRX;**************************************************TER00070
800010    1  0  DCL THISSVR PTR EXTERNAL STATIC;   /* -> CURRENT SVR, SET BY SAHER */ SVR00010
                                                                                  SVR00020
800030    1  0  DCL 1 SVR BASED (THISSVR), %INCLUDE SVR;**************************SVR00030
                    70 NEXT PTR,                                                  SVR00010
                    70 STIME FIXED BIN (31),                                      SVR00020
                    70 STIMEQ FIXED BIN (31),     /* ACCUMULATED STIME          */SVR00030
                    70 RTIME FIXED BIN (31),                                      SVR00040
                    70 RRATE FIXED BIN (15,7),    /* REAL CPU TICK RATE         */SVR00050
                    70 ID FIXED BIN                                               SVR00060
               ****************************************   ;                      SVR00030
```

```
 NUMBER LEV NT

                                                                                SVR00040
                 **************                                                 TER00070
                 %INCLUDE PFSVC;*************************************************TER00080
1100010   1  0   DCL PT_SVC PTR;                                                PFS00010
                                                                                PFS00020
1100030   1  0   DCL 1 PF_SVC BASED (PT_SVC),                                   PFS00030
                       2 SVC CHAR (7) VAR,                                      PFS00040
                       2 PTR PTR;                                               PFS00050
                                                                                PFS00060
                 ***************                                                TER00080
                 %INCLUDE MSG;***************************************************TER00090
1200010   1  0   DCL MSGLEN FIXED BIN;                                          MSG00010
1200020   1  0   DCL MSGPT PTR;                                                 MSG00020
1200030   1  0   DCL 1 MSG BASED (MSGPT),                                       MSG00030
                       2 LEN FIXED BIN,                                         MSG00040
                       2 STR CHAR (MSGLEN REFER (LEN));                         MSG00050
                                                                                MSG00060
                 *************                                                  TER00090
1200100   1  0   DCL (LEVEL,VPID,BOXID) FIXED BIN;                             TER00100
1200110   1  0   DCL MESSAGE CHAR(80) VAR;                                      TER00110
1200120   1  0   DCL COMMAND CHAR(12);                                          TER00120
1200130   1  0   DCL TYPE CHAR(8) VAR;                                          TER00130
                                                                                TER00140
1200150   1  0   LOOP:                                                         TER00150
                     PUT SKIP EDIT (' TERM: LEVEL = ',VP.LEVEL,' VP = ',VP.VPID,  TER00160
                         ' WAIT.BOX = ',VP.WAIT.BOX,' SVR = ',SVR.ID,' STIME = ',  TER00170
                         SVR.STIME) (A,F(2),A,F(2),A,F(2),A,F(2),A,F(11));      TER00180
1200190   1  0       MSGPT = VP.WAIT.MSG;                                       TER00190
1200200   1  0       IF (MSGPT ^= NULL) THEN DO;                                TER00200
1200210   1  1          PUT SKIP EDIT (' MSG: ',MSG.STR) (A,A(MSG.LEN));        TER00210
1200220   1  1          FREE MSG;                                               TER00220
1200230   1  1          VP.WAIT.MSG = NULL;                                     TER00230
1200240   1  1       END;                                                       TER00240
                                                                                TER00250
1200260   1  0   WORK:                                                         TER00260
                     PUT SKIP;                                                  TER00270
1200280   1  0       DISPLAY (' COMMAND?') REPLY (COMMAND);                     TER00280
                                                                                TER00290
1200300   1  0       IF (COMMAND = 'BUILD') THEN DO;                            TER00300
1200310   1  1          PUT SKIP LIST ('  LEVEL> ');                           TER00310
1200320   1  1          GET LIST (LEVEL);                                       TER00320
1200330   1  1          MSGLEN = 7;                                             TER00330
1200340   1  1          ALLOCATE MSG;                                           TER00340
1200350   1  1          MSG.STR = 'NEW VP!';                                    TER00350
1200360   1  1          ALLOCATE PF_SVC;                                        TER00360
1200370   1  1          PF_SVC.SVC = 'TERM';                                    TER00370
1200380   1  1          PF_SVC.PTR = MSGPT;                                     TER00380
1200390   1  1          CALL SEND (LEVEL,0,1,'S',15,PT_SVC);                    TER00390
1200400   1  1          GOTO WORK;                                              TER00400
```

PL/I OPTIMIZING COMPILER TERM: PROC;

```
NUMBER LEV NT

1200410   1  1      END;                                                          TER00410
                                                                                 TER00420
                                                                                 TER00430
1200430   1  0      ELSE IF (COMMAND = 'SEND') THEN DO;                           TER00440
1200440   1  1         PUT SKIP LIST ('  LEVEL,VPID,BOXID,TYPE,MESSAGE> ');      TER00450
1200450   1  1         GET LIST (LEVEL,VPID,BOXID,TYPE,MESSAGE);                  TER00460
1200460   1  1         MSGLEN = LENGTH (MESSAGE);                                 TER00470
1200470   1  1         ALLOCATE MSG;                                              TER00480
1200480   1  1         MSG.STR = MESSAGE;                                         TER00490
1200490   1  1         CALL SEND (LEVEL,VPID,BOXID,TYPE,MSGLEN,MSGPT);            TER00500
1200500   1  1         GOTO WORK;                                                 TER00510
1200510   1  1      END;                                                          TER00520
                                                                                 TER00530
1200530   1  0      ELSE IF (COMMAND = 'WAIT') THEN DO;                           TER00540
1200540   1  1         PUT SKIP LIST ('  BOX> ');                                TER00550
1200550   1  1         GET LIST (BOXID);                                          TER00560
1200560   1  1         PUT SKIP LIST ('  WAITING');                               TER00570
1200570   1  1         CALL WAIT (BOXID);                                         TER00580
1200580   1  1         GOTO LOOP;                                                 TER00590
1200590   1  1      END;                                                          TER00600
                                                                                 TER00610
1200610   1  0      ELSE IF (COMMAND = 'SYNC') THEN DO;                           TER00620
1200620   1  1         PUT SKIP LIST ('  SYNCING');                              TER00630
1200630   1  1         CALL SYNC;                                                 TER00640
1200640   1  1         GOTO LOOP;                                                 TER00650
1200650   1  1      END;                                                          TER00660
                                                                                 TER00670
1200670   1  0      ELSE IF (COMMAND = 'FINISH') THEN DO;                         TER00680
1200680   1  1         PUT SKIP LIST ('  FINISHING');                            TER00690
1200690   1  1         CALL FINISH;                                               TER00700
1200700   1  1         RETURN;                                                    TER00710
1200710   1  1      END;                                                          TER00720
                                                                                 TER00730
1200730   1  0      ELSE DO; /* UNKNOWN COMMAND */                              TER00740
1200740   1  1         PUT LIST ('  ??');                                        TER00750
1200750   1  1         GOTO WORK;                                                 TER00760
1200760   1  1      END;                                                          TER00770
                                                                                 TER00780
1200780   1  0   END TERM;
```

SOURCE LISTING

```
NUMBER LEV NT                                                                    R

    10      0  VPER: PROC (VP);                                                  VPE00010
                                                                                 VPE00020
    30   1  0  DCL (THIS, NEXT) PTR;                                            VPE00030
    40   1  0  DCL THISVP PTR EXTERNAL STATIC;                                  VPE00040
    50   1  0  DCL 1 VP,                                                        VPE00050
                   80 BDSA FIXED BIN (31),      /* ADDR (BOTTOM DSA)          */VP 00010
                   80 TDSA FIXED BIN (31),      /* ADDR (TOP DSA)             */VP 00020
                   80 SAVESIZE FIXED BIN (31),  /* SIZE OF AREA TO BE SAVED   */VP 00030
                   80 SAFE PTR,                 /* -> 1ST SAVBLK              */VP 00040
                   80 PROCNAME CHAR (7) VAR,    /* NAME OF TOP LEVEL PROC     */VP 00050
                   80 STATUS CHAR (12) VAR,                                      VP 00060
                   80 MAIL PTR,                 /* CHAIN OF INCOMING MAIL     */VP 00070
                   80 WAIT,                                                      VP 00080
                      81 BOX FIXED BIN,         /* BOX AWAITING MAIL          */VP 00090
                      81 MSG PTR,               /* MSG IN WAIT.BOX            */VP 00100
                   80 VPID FIXED BIN,           /* INDEX IN THE VPST          */VP 00110
                   80 LEVEL FIXED BIN,          /* LEVEL                      */VP 00120
                   80 VTIME FIXED BIN (31)                                       VP 00130
                  ;                                                              VPE00050
300010   1  0  DCL THISSVR PTR EXTERNAL STATIC;  /* -> CURRENT SVR, SET BY SAHER */ SVR00010
                                                                                 SVR00020
300030   1  0  DCL 1 SVR BASED (THISSVR),                                        SVR00030
                   70 NEXT PTR,                                                  SVR00010
                   70 STIME FIXED BIN (31),                                      SVR00020
                   70 STIMEQ FIXED BIN (31),    /* ACCUMULATED STIME          */SVR00030
                   70 RTIME FIXED BIN (31),                                      SVR00040
                   70 RRATE FIXED BIN (15,7),   /* REAL CPU TICK RATE         */SVR00050
                   70 ID FIXED BIN                                               SVR00060
                  ;                                                              SVR00030
600010   1  0  DCL BOXPT PTR;                                                    BOX00010
                                                                                 BOX00020
600030   1  0  DCL 1 BOX BASED (BOXPT),                                         BOX00030
                   2 NEXT PTR,                                                   BOX00040
                   2 ID FIXED BIN,                                               BOX00050
                   2 LIST PTR;                                                   BOX00060
                                                                                 SAV00030
                                                                                 SAV00060
700070   1  0  DCL NEWSAVBLK PTR;                                                SAV00070
                                                                                 SAV00080
700090   1  0  DCL 1 SAVBLK BASED (NEWSAVBLK),                                   SAV00090
                   2 NEXT PTR,                  /* NEXT SAVED ITEM            */SAV00100
                   2 ADDR FIXED BIN (31),       /* BEG ADDR OF SAVED ITEM     */SAV00110
                   2 SIZE FIXED BIN (31),       /* SIZE OF SAVED ITEM         */SAV00120
                   2 AREA (     250) FIXED BIN (31);                             SAV00130  1
                                                                                 SAV00140
                                                                                 VPE00090
```

```
 NUMBER LEV NT                                                                    R

 900010   1  0  DCL EXECUTE ENTRY (CHAR(*) VAR);                                  EXE00010
1000010   1  0  DCL DSA FIXED BIN (31,0),                                        GET00010
                    NAB FIXED BIN (31,0),                                       GET00020
                    SEG FIXED BIN (31,0),                                       GET00030
                    EOS FIXED BIN (31,0);                                       GET00040
                                                                                GET00050
                                                                                GET00060
1000070   1  0  DCL GET4 ENTRY (FIXED BIN (31), FIXED BIN (31),                 GET00070
                              FIXED BIN (31), FIXED BIN (31))                   GET00080
                         OPTIONS (ASSEMBLER, INTER);                            GET00090
1100010   1  0  DCL RTSTK ENTRY (PTR, FIXED BIN (31))                          RTS00010
                          OPTIONS (ASSEMBLER, INTER);                           RTS00020
                                                                                RTS00030
                    /* ARG1: PTR -> SAVBLK                                      RTS00040
                       ARG2: ADDR OF RETURN DSA (TOP DSA)                       RTS00050
                    */                                                          RTS00060
1100130   1  0  DCL (ADDR,NULL) BUILTIN;                                        VPE00130
1200010   1  0  DCL STIMER ENTRY RETURNS (FIXED BIN (31));                      STI00010
                                                                                VPE00150
                                                                                VPE00160
1200170   1  0     THISVP = ADDR(VP);                                           VPE00170
                                                                                VPE00180
1200190   1  0     IF VP.STATUS = 'NASCENT' THEN DO; /* STARTING A VP */        VPE00190
                                                                                VPE00200
1200210   1  1        CALL GET4 (DSA,NAB,SEG,EOS);                             VPE00210
1200220   1  1        IF SEG ^= 255 THEN DO;                                   VPE00220
1200230   1  2           PUT SKIP LIST (' ERROR: ILLEGAL SEGNO (VPER)');       VPE00230
1200240   1  2           PUT DATA (SEG);                                       VPE00240
1200250   1  2           STOP;                                                 VPE00250
1200260   1  2        END;                                                     VPE00260
                                                                                VPE00270
1200280   1  1        VP.BDSA = DSA;                                           VPE00280
1200290   1  1        VP.STATUS = 'RUNNING'; /* SET TO RUNNING */              VPE00290
1200300   1  1        CALL EXECUTE (VP.PROCNAME);                              VPE00300
                                                                                VPE00310
                       /*** SVSTK & RTSTK SHOULD RETURN HERE TOO ***/          VPE00320
                                                                                VPE00330
1200340   1  1        VP.VTIME = VP.VTIME + (STIMER - SVR.STIME);              VPE00340
1200350   1  1        IF VP.WAIT.BOX = -1 THEN DO; /* FINISHED */              VPE00350
1200360   1  2           THIS = VP.SAFE;                                       VPE00360
1200370   1  2           DO WHILE (THIS ^= NULL);                              VPE00370
1200380   1  3              NEXT = THIS -> SAVBLK.NEXT;                        VPE00380
1200390   1  3              FREE THIS -> SAVBLK;                               VPE00390
1200400   1  3              THIS = NEXT;                                       VPE00400
1200410   1  3           END;                                                  VPE00410
1200420   1  2           VP.SAFE = NULL;                                       VPE00420
1200430   1  2           THIS = VP.MAIL;                                       VPE00430
1200440   1  2           DO WHILE (THIS ^= NULL);                              VPE00440
1200450   1  3              NEXT = THIS -> BOX.NEXT;                           VPE00450
```

```
 NUMBER LEV NT                                                                                    R

1200460   1  3              FREE THIS -> BOX;                                                    VPE00460
1200470   1  3              THIS = NEXT;                                                         VPE00470
1200480   1  3           END;                                                                    VPE00480
1200490   1  2           VP.MAIL = NULL;                                                         VPE00490
1200500   1  2        END; /* FREEING */                                                        VPE00500

1200520   1  1        RETURN;                                                                   VPE00510
1200530   1  1     END;                                                                         VPE00520

1200550   1  0     ELSE DO; /* STATUS = RUNNABLE */                                            VPE00530
                                                                                               VPE00540
1200570   1  1        CALL GET4 (DSA,NAB,SEG,EOS);                                            VPE00550
1200580   1  1        IF SEG ^= 255 THEN DO;                                                  VPE00560
1200590   1  2           PUT SKIP LIST (' ERROR: ILLEGAL SEGNO (VPER)');                      VPE00570
1200600   1  2           PUT DATA (SEG);                                                      VPE00580
1200610   1  2           STOP;                                                                VPE00590
1200620   1  2        END;                                                                    VPE00600
1200630   1  1        IF (EOS-DSA) < (VP.SAVESIZE) THEN DO;                                   VPE00610
1200640   1  2           PUT SKIP LIST                                                        VPE00620
                              (' ERROR: NOT ENOUGH SPACE TO RESUME VP (VPER)');              VPE00630
1200660   1  2           PUT DATA (VP.VPID,VP.SAVESIZE,DSA,EOS);                              VPE00640
1200670   1  2           STOP;                                                                VPE00650
1200680   1  2        END;                                                                    VPE00660
1200690   1  1        IF DSA ^= VP.BDSA THEN DO;                                              VPE00670
1200700   1  2           PUT SKIP LIST (' ERROR: BDSA DISCREPENCY (VPER)');                   VPE00680
1200710   1  2           PUT DATA (VP.VPID,VP.BDSA,DSA);                                      VPE00690
1200720   1  2           STOP;                                                                VPE00700
1200730   1  2        END;                                                                    VPE00710
                                                                                               VPE00720
1200750   1  1        VP.STATUS = 'RUNNING';                                                  VPE00730
1200760   1  1        CALL RTSTK (VP.SAFE,VP.TDSA);                                           VPE00740
                                                                                               VPE00750
                        /* CONTROL SHOULD NEVER REACH HERE */                                  VPE00760
1200790   1  1        PUT SKIP LIST (' ERROR: ILLEGAL RETURN FROM RTSTK (VPER)');            VPE00770
1200800   1  1        STOP;                                                                   VPE00780

1200820   1  1     END; /* RESUMING A VP */                                                   VPE00790
                                                                                               VPE00800
1200840   1  0  END VPER;                                                                     VPE00810
                                                                                               VPE00820
                                                                                               VPE00830
                                                                                               VPE00840
```

SOURCE LISTING

```
NUMBER LEV NT

    10      0  VPSTART: PROC (VP, PROCNAME, ARGPT);                                  VPS00010
                                                                                      VPS00020
    30   1  0  DCL 1 VP, %INCLUDE VP;*****************************************VPS00030
                    80 BDSA FIXED BIN (31),       /* ADDR (BOTTOM DSA)          */VP 00010
                    80 TDSA FIXED BIN (31),       /* ADDR (TOP DSA)             */VP 00020
                    80 SAVESIZE FIXED BIN (31),   /* SIZE OF AREA TO BE SAVED   */VP 00030
                    80 SAFE PTR,                  /* -> 1ST SAVBLK              */VP 00040
                    80 PROCNAME CHAR (7) VAR,     /* NAME OF TOP LEVEL PROC     */VP 00050
                    80 STATUS CHAR (12) VAR,                                        VP 00060
                    80 MAIL PTR,                  /* CHAIN OF INCOMING MAIL     */VP 00070
                    80 WAIT,                                                        VP 00080
                       81 BOX FIXED BIN,          /* BOX AWAITING MAIL          */VP 00090
                       81 MSG PTR,                /* MSG IN WAIT.BOX            */VP 00100
                    80 VPID FIXED BIN,            /* INDEX IN THE VPST          */VP 00110
                    80 LEVEL FIXED BIN,           /* LEVEL                      */VP 00120
                    80 VTIME FIXED BIN (31)                                         VP 00130
                ***********************   ;                                       VPS00030
 200040   1  0  DCL PROCNAME CHAR(*) VAR;                                          VPS00040
 200050   1  0  DCL ARGPT PTR;                                                     VPS00050
                                                                                  VPS00060
 200070   1  0  DCL NULL BUILTIN;                                                 VPS00070
                %INCLUDE QEVENT;*************************************************VPS00080
 300010   1  0  DCL QEVENT ENTRY (FIXED BIN, FIXED BIN (31), CHAR(*) VAR,        QEV00010
                              FIXED BIN (31), PTR);                               QEV00020
                ****************                                                  VPS00080
                %INCLUDE STIMER;*************************************************VPS00090
 400010   1  0  DCL STIMER ENTRY RETURNS (FIXED BIN (31));                        STI00010
                ****************                                                  VPS00090
                                                                                  VPS00100
 400110   1  0      VP.STATUS = 'NASCENT';                                        VPS00110
 400120   1  0      VP.PROCNAME = PROCNAME;                                       VPS00120
 400130   1  0      VP.SAFE = NULL;                                               VPS00130
 400140   1  0      VP.WAIT.BOX = 0;                                              VPS00140
 400150   1  0      VP.WAIT.MSG = ARGPT;                                          VPS00150
 400160   1  0      VP.SAVESIZE = 0;                                              VPS00160
 400170   1  0      VP.MAIL = NULL;                                               VPS00170
 400180   1  0      VP.VTIME = 0;                                                 VPS00180
 400190   1  0      CALL QEVENT (VP.LEVEL*3+3,STIMER,'SYNC',VP.VPID,NULL);        VPS00190
                    /* KICK START */                                              VPS00200
                                                                                  VPS00210
 400220   1  0  END VPSTART;                                                      VPS00220
```

SOURCE LISTING

```
NUMBER LEV NT

    10      0  WAIT: PROC (BOXID);                                                    WAI00010
                                                                                     WAI00020
               %INCLUDE QEVENT;*************************************************WAI00030
100010    1  0  DCL QEVENT ENTRY (FIXED BIN, FIXED BIN (31), CHAR(*) VAR,          QEV00010
                        FIXED BIN (31), PTR);                                      QEV00020
               ****************                                                      WAI00030
               %INCLUDE STIMER;*************************************************WAI00040
200010    1  0  DCL STIMER ENTRY RETURNS (FIXED BIN (31));                         STI00010
               ****************                                                      WAI00040
200050    1  0  DCL SLEEP ENTRY;                                                   WAI00050
200060    1  0  DCL (NULL) BUILTIN;                                                WAI00060
                                                                                   WAI00070
               %INCLUDE VPX;****************************************************WAI00080
300010    1  0  DCL THISVP PTR EXTERNAL STATIC;        /* -> CURRENT VP, SET BY VPER */VPX00010
                                                                                   VPX00020
300030    1  0  DCL 1 VP BASED (THISVP), %INCLUDE VP;*********************************VPX00030
                    80 BDSA FIXED BIN (31),       /* ADDR (BOTTOM DSA)          */VP 00010
                    80 TDSA FIXED BIN (31),       /* ADDR (TOP DSA)             */VP 00020
                    80 SAVESIZE FIXED BIN (31),   /* SIZE OF AREA TO BE SAVED   */VP 00030
                    80 SAFE PTR,                  /* -> 1ST SAVBLK              */VP 00040
                    80 PROCNAME CHAR (7) VAR,     /* NAME OF TOP LEVEL PROC     */VP 00050
                    80 STATUS CHAR (12) VAR,                                       VP 00060
                    80 MAIL PTR,                  /* CHAIN OF INCOMING MAIL     */VP 00070
                    80 WAIT,                                                       VP 00080
                       81 BOX FIXED BIN,          /* BOX AWAITING MAIL          */VP 00090
                       81 MSG PTR,                /* MSG IN WAIT.BOX            */VP 00100
                    80 VPID FIXED BIN,            /* INDEX IN THE VPST          */VP 00110
                    80 LEVEL FIXED BIN,           /* LEVEL                      */VP 00120
                    80 VTIME FIXED BIN (31)                                        VP 00130
               ************************************    ;                           VPX00030
                                                                                   VPX00040
               *************                                                       WAI00080
500090    1  0  DCL BOXID FIXED BIN;                                              WAI00090
                                                                                   WAI00100
500110    1  0      VP.WAIT.BOX = BOXID;                                          WAI00110
500120    1  0      CALL QEVENT (VP.LEVEL*3+3,STIMER,'WAIT',VP.VPID,NULL);        WAI00120
500130    1  0      CALL SLEEP;                                                   WAI00130
                                                                                   WAI00140
500150    1  0  END WAIT;                                                         WAI00150
```

APPENDIX B: LISTINGS OF ASSEMBLY LANGUAGE ROUTINES

ASSEMBLY LANGUAGE ROUTINES:

GET4
RTIMER
RTSTK
SVSTK

```
 LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                          GET00010
                                    1 * GET4 GETS THE VALUE OF THE CURRENT DSA, NAB, SEGNO & EOS  GET00020
                                    2 *                                                           GET00030
                                    3 *    DCL GET ENTRY OPTIONS (ASSEMBLE,INTER);                GET00040
                                    4 *    CALL GET4 (DSA,NAB,SEGNO,EOS);                         GET00050
                                    5 *                                                           GET00060
000000                              6 GT       CSECT                                              GET00070
                                    7          ENTRY GET4                                         GET00080
000000 C7C5E3F440                   8          DC    C'GET4 '                                     GET00090
000005 05                           9          DC    AL1(5)                                       GET00100
000006 90EC D00C      0000C        10 GET4    STM   14,12,12(13)   SAVE REGISTORS               GET00110
00000A 5861 0000      00000        11         L     6,0(1)                                       GET00120
00000E 50D6 0000      00000        12         ST    13,0(6)        ARG1 <= R13                   GET00130
000012 5861 0004      00004        13         L     6,4(1)                                       GET00140
000016 587D 004C      0004C        14         L     7,76(13)       R7 <= (SEGNO,NAB)             GET00150
00001A 4177 0000      00000        15         LA    7,0(7)         R7 <= NAB                     GET00160
00001E 5076 0000      00000        16         ST    7,0(6)         ARG2 <= NAB                   GET00170
000022 5861 0008      00008        17         L     6,8(1)                                       GET00180
000026 587D 004C      0004C        18         L     7,76(13)       R7 <= (SEGNO,NAB)             GET00190
00002A 8870 0018      00018        19         SRL   7,24           R7 <= SEGNO                   GET00200
00002E 5076 0000      00000        20         ST    7,0(6)         ARG3 <= SEGNO                 GET00210
000032 5861 000C      0000C        21         L     6,12(1)                                      GET00220
000036 587C 000C      0000C        22         L     7,12(12)       R7 <= (SEGNO,EOS)             GET00230
00003A 4177 0000      00000        23         LA    7,0(7)         R7 <= EOS                     GET00240
00003E 5076 0000      00000        24         ST    7,0(6)         ARG4 <= EOS                   GET00250
000042 98EC D00C      0000C        25         LM    14,12,12(13)   SAVE REGISTOR                 GET00260
000046 07FE                        26         BR    14                                           GET00270
                                   27         END
```

ASM 0201 21.34 05/18/81

```
 LOC  OBJECT CODE     ADDR1 ADDR2  STMT   SOURCE STATEMENT                                                    RTI00010
                                     1 RTM      CSECT                                                         RTI00020
                                     2          ENTRY RTIMER                                                  RTI00030
                                     3          DC    C' RTIMER'                                              RTI00040
000000                               4          DC    AL1(6)                                                  RTI00050
                                     5 RTIMER   STM   14,12,12(13)                                            RTI00060
000000 40D9E3C9D4C5D9                6          BALR  11,0                                                    RTI00070
000007 06                            7          USING *,11                                                    RTI00080
000008 90EC D00C       0000C         8          LA    4,BLK                                                   RTI00090
00000C 0580                    0000E 9          DC    X'8340000C'    DIAG R4,X'0C', GET BLOCK                 RTI00100
                       00030         10         L     4,28(4)        LOWER 32 BITS OF TOTAL CPU TIME          RTI00110
00000E 4140 B022                     11         N     4,=X'7FFFFFFF'                                          RTI00120
000012 8340000C                      12         L     1,0(1)                                                  RTI00130
000016 5844 001C       0001C         13         ST    4,0(1)                                                  RTI00140
00001A 5440 B042       00050         14         LM    14,12,12(13)                                            RTI00150
00001E 5811 0000       00000         15         BR    14                                                      RTI00160
000022 5041 0000       00000         16 *                                                                     RTI00170
000026 98EC D00C       0000C         17 BLK     DS    4D                                                      RTI00180
00002A 07FE                          18         END                                                           RTI00190
                                     19                =X'7FFFFFFF'

000030

000050 7FFFFFFF
```

```
LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                  ASM 0201 21.34 05/18/81
                                    1 * RTSTK RESTORES THE STACK:                                                  RTS00010
                                    2 *                                                                            RTS00020
                                    3 *   DCL RTSTK ENTRY OPTIONS (ASSEMBLE, INTER);                               RTS00030
                                    4 *   CALL RTSTK (PTR, DSA);                                                   RTS00040
                                    5 *                                                                            RTS00050
                                    6 *    WHERE                                                                   RTS00060
                                    7 *                                                                            RTS00070
                                    8 *   PTR -> SAVBLK: 0(SAVBLK) -> NEXT SAVBLK                                  RTS00080
                                    9 *                  4(SAVBLK) -> 1ST LOCATION TO SAVE/RESTORE                 RTS00090
                                   10 *                  8(SAVBLK) -> SIZE OF SAVBLK                               RTS00100
                                   11 *                 12(SAVBLK)... SAVE AREA                                    RTS00110
                                   12 *                                                                            RTS00120
                                   13 *   DSA -> RETURN DSA (WHERE RTSTK SHOULD RETURN)                            RTS00130
                                   14 *                                                                            RTS00140
000000                             15 RT       CSECT                                                               RTS00150
                                   16          ENTRY RTSTK                                                         RTS00160
000000 D9E3E2E3D2                  17          DC    C'RTSTK'                                                      RTS00170
000005 05                          18          DC    AL1(5)                                                        RTS00180
                                   19 *                                                                            RTS00190
000006 90EC D00C       0000C       20 RTSTK    STM   14,12,12(13)   SAVE REGISTORS                                RTS00200
00000A 05B0                        21          BALR  11,0          ESTABLISH BASE                                RTS00210
                             0000C 22          USING *,11                                                          RTS00220
00000C 186D                        23          LR    6,13          R6  -> OLD DSA                                RTS00230
00000E 41D0 B05C       00068       24          LA    13,SAFE       R13 -> NEW DSA                                RTS00240
000012 506D 0004       00004       25          ST    6,4(13)       CHAIN BACK                                    RTS00250
000016 9280 D000       00000       26          MVI   0(13),X'80'   SET FLAGS                                     RTS00260
00001A 9200 D001       00001       27          MVI   1(13),X'00'                                                 RTS00270
00001E D208 D048 6048  00048 00048 28          MVC   72(12,13),72(6) COPY A(LSW), ETC.                           RTS00280
                                   29 *                                                                            RTS00290
000024 5861 0000       00000       30          L     6,0(1)        R6 -> 1ST ARG -> 1ST SAVBLK                   RTS00300
000028 5866 0000       00000       31          L     6,0(6)        R6 -> 1ST SAVBLK                              RTS00310
                                   32 *                                                                            RTS00320
00002C 5878 0004       00004       33 AGAIN    L     7,4(6)        R7 -> SAVBLK.ADDR                             RTS00330
000030 4186 000C       0000C       34          LA    8,12(6)       R8 -> SAVBLK.AREA                             RTS00340
000034 58A6 0008       00008       35          L     10,8(6)       R10 -> SAVBLK.SIZE                            RTS00350
000038 58A0 B25C       00268       36          S     10,=F'4'      WE ARE COUNTING FROM SIZE-4TO 0               RTS00360
                                   37 *                                                                            RTS00370
00003C 589A 8000       00000       38 LOOP     L     9,0(10,8)     GET WORD                                      RTS00380
000040 509A 7000       00000       39          ST    9,0(10,7)     RESTORE WORD                                  RTS00390
000044 58A0 B25C       00268       40          S     10,=F'4'      DEC COUNT                                     RTS00400
000048 47B0 B030       0003C       41          BNM   LOOP          AGAIN IF MORE                                 RTS00410
                                   42 *                                                                            RTS00420
00004C 5866 0000       00000       43          L     6,0(6)        R6 -> SAVBLK.NEXT                             RTS00430
000050 5560 B260       0026C       44          CL    6,=X'FF000000' NULL?                                        RTS00440
000054 4770 B020       0002C       45          BNE   AGAIN                                                       RTS00450
                                   46 *                                                                            RTS00460
000058 5861 0004       00004       47          L     6,4(1)        R6 -> 2ND ARG -> TOP DSA                      RTS00470
00005C 58D6 0000       00000       48          L     13,0(6)       R13 -> TOP DSA                                RTS00480
000060 98EC D00C       0000C       49          LM    14,12,12(13)  RETURN THERE                                  RTS00490
000064 07FE                        50          BR    14                                                          RTS00500
                                   51 *                                                                            RTS00510
000068                             52 SAFE     DS    120F                                                        RTS00520
                                   53          END                                                               RTS00530
000268 00000004                    54                =F'4'
00026C FF000000                    55                =X'FF000000'
```

```
 LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                    ASM 0201 21.34 05/18/81
                                     1 * SVSTK SAVES THE STACK:                                                    SVS00010
                                     2 *                                                                           SVS00020
                                     3 *   DCL SVSTK ENTRY OPTIONS (ASSEMBLE, INTER);                              SVS00030
                                     4 *   CALL SVSTK (PTR, DSA);                                                  SVS00040
                                     5 *                                                                           SVS00050
                                     6 *    WHERE                                                                  SVS00060
                                     7 *                                                                           SVS00070
                                     8 *   PTR -> SAVBLK: 0(SAVBLK) -> NEXT SAVBLK                                 SVS00080
                                     9 *                  4(SAVBLK) -> 1ST LOCATION TO SAVE/RESTORE                SVS00090
                                    10 *                  8(SAVBLK) -> SIZE OF SAVBLK                              SVS00100
                                    11 *                 12(SAVBLK)... SAVE AREA                                   SVS00110
                                    12 *                                                                           SVS00120
                                    13 *   DSA -> RETURN DSA (WHERE SVSTK SHOULD RETURN)                           SVS00130
                                    14 *                                                                           SVS00140
000000                              15 SV       CSECT                                                              SVS00150
                                    16          ENTRY SVSTK                                                        SVS00160
000000 E2E5E2E3D2                   17          DC    C'SVSTK'                                                     SVS00170
000005 05                           18          DC    AL1(5)                                                       SVS00180
                                    19 *                                                                           SVS00190
000006 90EC D00C      0000C         20 SVSTK    STM   14,12,12(13)   SAVE REGISTORS                               SVS00200
00000A 05B0                         21          BALR  11,0           ESTABLISH BASE                               SVS00210
                            0000C   22          USING *,11                                                         SVS00220
00000C 186D                         23          LR    6,13           R6  -> OLD DSA                               SVS00230
00000E 41D0 B05C      00068         24          LA    13,SAFE        R13 -> NEW DSA                               SVS00240
000012 506D 0004      00004         25          ST    6,4(13)        CHAIN BACK                                   SVS00250
000016 9280 D000      00000         26          MVI   0(13),X'80'    SET FLAGS                                    SVS00260
00001A 9200 D001      00001         27          MVI   1(13),X'00'                                                 SVS00270
00001E D20B D048 6048 00048 00048   28          MVC   72(12,13),72(6) COPY A(LSW), ETC.                           SVS00280
                                    29 *                                                                           SVS00290
000024 5861 0000      00000         30          L     6,0(1)         R6 -> 1ST ARG -> 1ST SAVBLK                  SVS00300
000028 5866 0000      00000         31          L     6,0(6)         R6 -> 1ST SAVBLK                             SVS00310
                                    32 *                                                                           SVS00320
00002C 5876 0004      00004         33 AGAIN    L     7,4(6)         R7 -> SAVBLK.ADDR                            SVS00330
000030 4186 000C      0000C         34          LA    8,12(6)        R8 -> SAVBLK.AREA                            SVS00340
000034 58A6 0008      00008         35          L     10,8(6)        R10 -> SAVBLK.SIZE                           SVS00350
000038 5BA0 B25C      00268         36          S     10,=F'4'       WE ARE COUNTING FROM SIZE-4 TO 0             SVS00360
                                    37 *                                                                           SVS00370
00003C 589A 7000      00000         38 LOOP     L     9,0(10,7)      GET WORD                                     SVS00380
000040 509A 8000      00000         39          ST    9,0(10,8)      RESTORE WORD                                 SVS00390
000044 5BA0 B25C      00268         40          S     10,=F'4'       DEC COUNT                                    SVS00400
000048 47B0 B030      0003C         41          BNM   LOOP           AGAIN IF MORE                                SVS00410
                                    42 *                                                                           SVS00420
00004C 5866 0000      00000         43          L     6,0(6)         R6 -> SAVBLK.NEXT                            SVS00430
000050 5560 B260      0026C         44          CL    6,=X'FF000000' NULL?                                        SVS00440
000054 4770 B020      0002C         45          BNE   AGAIN                                                       SVS00450
                                    46 *                                                                           SVS00460
000058 5861 0004      00004         47          L     6,4(1)         R6 -> 2ND ARG -> TOP DSA                     SVS00470
00005C 58D6 0000      00000         48          L     13,0(6)        R13 -> TOP DSA                               SVS00480
000060 98EC D00C      0000C         49          LM    14,12,12(13)   RETURN THERE                                 SVS00490
000064 07FE                         50          BR    14                                                         SVS00500
                                    51 *                                                                           SVS00510
000068                              52 SAFE     DS    128F                                                       SVS00520
                                    53          END                                                              SVS00530
000268 00000004                     54                =F'4'
00026C FF000000                     55                =X'FF000000'
```

APPENDIX C: LISTINGS OF MACRO FILES (DATA DECLARATIONS)

MACRO FILES:

BOX
CONFIG
DEBUG
EVENT
LIST
LOS
LOSX
MSG
PARAMS
PFADDR
PFLEVEL
PFMSG
PFS
PFSVC
QUEUES
SAVBLK
SVR
SVRX
VP
VPX

```
DCL BOXPT PTR;                                          BOX00010
                                                        BOX00020
DCL 1 BOX BASED (BOXPT),                                BOX00030
      2 NEXT PTR,                                       BOX00040
      2 ID FIXED BIN,                                   BOX00050
      2 LIST PTR;                                       BOX00060
                                                        BOX00070
```

```
%DECLARE MAXLEVEL FIXED;                                    CON00010
%DECLARE MAXQ FIXED;                                        CON00020
%MAXLEVEL = 1;                                              CON00030
%MAXQ = 6;                                                  CON00040
```

```
DCL 1 DEBUG EXTERNAL STATIC,                                    DEB00010
      2 SLEEP$ BIT(1) INIT ('0'B),                              DEB00020
      2 SAHER$ BIT(1) INIT ('0'B),                              DEB00030
      2 SCHEDULER$ BIT(1) INIT ('0'B),                          DEB00040
      2 SHELL$ BIT(1) INIT ('0'B),                              DEB00050
      2 AAHER$ BIT(1) INIT ('0'B);                              DEB00060
```

```
      70 NEXT PTR,                    /* NEXT EVENT              */ EVE00010
      70 STIME FIXED BIN (31),                                      EVE00020
      70 TYPE CHAR (12) VAR,                                        EVE00030
      70 INDEX FIXED BIN (31),                                      EVE00040
      70 PTR PTR                                                    EVE00050
```

```
DCL LISTPT PTR;                                          LIS00010
                                                         LIS00020
DCL 1 LIST BASED (LISTPT),                               LIS00030
      2 NEXT PTR,                                        LIS00040
      2 THIS PTR;                                        LIS00050
                                                         LIS00060
```

```
      70 LEVEL FIXED BIN,                                          LOS00010
      70 SRQ,                                                      LOS00020
         71 SIZE FIXED BIN,                                        LOS00030
         71 MAX FIXED BIN,                                         LOS00040
      70 DBB,                                                      LOS00050
         71 SIZE FIXED BIN,                                        LOS00060
         71 MAX FIXED BIN,                                         LOS00070
      70 VPS, %DECLARE MAXVP FIXED; %MAXVP = 20;                   LOS00080
         71 TABLE (MAXVP), %INCLUDE VP;                            LOS00090
```

```
DCL THISLOS PTR EXTERNAL STATIC;     /* -> CURRENT LOS, ST BY SAHER */LOS00010
                                                                     LOS00020
DCL 1 LOS BASED (THISLOS), %INCLUDE LOS;   ;                         LOS00030
                                                                     LOS00040
```

```
DCL MSGLEN FIXED BIN;                                    MSG00010
DCL MSGPT PTR;                                           MSG00020
DCL 1 MSG BASED (MSGPT),                                 MSG00030
      2 LEN FIXED BIN,                                   MSG00040
      2 STR CHAR (MSGLEN REFER (LEN));                   MSG00050
                                                         MSG00060
```

```
DCL 1 PARAMS EXTERNAL STATIC,                                  PAR00010
      2 THRU_RATE FIXED BIN (31,7),                            PAR00020
      2 DELAY_GB_GC FIXED BIN (31),                            PAR00030
      2 DELAY_GC_GB FIXED BIN (31),                            PAR00040
      2 TERMINALS FIXED BIN;                                   PAR00050
                                                               PAR00060
```

```
DCL PT_ADDR PTR;                                        PFA00010
                                                        PFA00020
DCL 1 PF_ADDR BASED (PT_ADDR),                          PFA00030
      2 VPID FIXED BIN,                                 PFA00040
      2 BOXID FIXED BIN,                                PFA00050
      2 PTR PTR;                                        PFA00060
                                                        PFA00070
```

```
DCL PT_LEVEL PTR;                                          PFL00010
                                                           PFL00020
DCL 1 PF_LEVEL BASED (PT_LEVEL),                           PFL00030
      2 LEVEL FIXED BIN,                                   PFL00040
      2 PTR PTR;                                           PFL00050
                                                           PFL00060
```

```
DCL PT_MSG PTR;                                          PFM00010
                                                         PFM00020
DCL 1 PF_MSG BASED (PT_MSG),                             PFM00030
      2 LEN FIXED BIN,                                   PFM00040
      2 TYPE CHAR (12) VAR,                              PFM00050
      2 PTR PTR;                                         PFM00060
                                                         PFM00070
```

FILE: PFS PLI A

CONVERSATIONAL MONITOR SYSTEM

```
                                                        PFS00010
DCL PT_S PTR;                                           PFS00020
                                                        PFS00030
DCL 1 PF_S BASED (PT_S),                                PFS00040
      2 LEN FIXED BIN,                                  PFS00050
      2 PTR PTR;                                        PFS00060
```

```
DCL PT_SVC PTR;                                          PFS00010
                                                         PFS00020
DCL 1 PF_SVC BASED (PT_SVC),                             PFS00030
      2 SVC CHAR (7) VAR,                                PFS00040
      2 PTR PTR;                                         PFS00050
                                                         PFS00060
```

CONVERSATIONAL MONITOR SYSTEM

FILE: QUEUES PLI A

```
DCL SQUEUE (0:MAXQ) PTR EXTERNAL STATIC;                QS 00010
DCL EQUEUE (0:MAXQ) PTR EXTERNAL STATIC;                QS 00020
                                                        QS 00030
```

```
%DECLARE MAXSAVBYTE FIXED;                                          SAV00010
%DECLARE MAXSAVWORD FIXED;                                          SAV00020
                                                                    SAV00030
%MAXSAVBYTE = 1000;                                                 SAV00040
%MAXSAVWORD =  250;                                                 SAV00050
                                                                    SAV00060
DCL NEWSAVBLK PTR;                                                  SAV00070
                                                                    SAV00080
DCL 1 SAVBLK BASED (NEWSAVBLK),                                     SAV00090
      2 NEXT PTR,                  /* NEXT SAVED ITEM            */SAV00100
      2 ADDR FIXED BIN (31),       /* BEG ADDR OF SAVED ITEM     */SAV00110
      2 SIZE FIXED BIN (31),       /* SIZE OF SAVED ITEM         */SAV00120
      2 AREA (MAXSAVWORD) FIXED BIN (31);                           SAV00130
                                                                    SAV00140
%DEACTIVATE MAXSAVWORD;                                             SAV00150
                                                                    SAV00160
```

```
      70 NEXT PTR,                                                SVR00010
      70 STIME FIXED BIN (31),                                    SVR00020
      70 STIMEQ FIXED BIN (31),       /* ACCUMULATED STIME       */SVR00030
      70 RTIME FIXED BIN (31),                                    SVR00040
      70 RRATE FIXED BIN (15,7),      /* REAL CPU TICK RATE      */SVR00050
      70 ID FIXED BIN                                             SVR00060
```

```
DCL THISSVR PTR EXTERNAL STATIC;   /* -> CURRENT SVR, SET BY SAHER */  SVR00010
                                                                      SVR00020
DCL 1 SVR BASED (THISSVR), %INCLUDE SVR;   ;                          SVR00030
                                                                      SVR00040
```

```
      80 BDSA FIXED BIN (31),          /* ADDR (BOTTOM DSA)                */VP 00010
      80 TDSA FIXED BIN (31),          /* ADDR (TOP DSA)                   */VP 00020
      80 SAVESIZE FIXED BIN (31),      /* SIZE OF AREA TO BE SAVED         */VP 00030
      80 SAFE PTR,                     /* -> 1ST SAVBLK                    */VP 00040
      80 PROCNAME CHAR (7) VAR,        /* NAME OF TOP LEVEL PROC           */VP 00050
                                                                             VP 00060
      80 STATUS CHAR (12) VAR,         /* CHAIN OF INCOMING MAIL           */VP 00070
      80 MAIL PTR,                                                           VP 00080
      80 WAIT,                         /* BOX AWAITING MAIL                */VP 00090
         81 BOX FIXED BIN,             /* MSG IN WAIT.BOX                  */VP 00100
         81 MSG PTR,                   /* INDEX IN THE VPST                */VP 00110
      80 VPID FIXED BIN,               /* LEVEL                            */VP 00120
      80 LEVEL FIXED BIN,                                                    VP 00130
      80 VTIME FIXED BIN (31)
```

```
DCL THISVP PTR EXTERNAL STATIC;        /* -> CURRENT VP, SET BY VPER */VPX00010
                                                                      VPX00020
DCL 1 VP BASED (THISVP), %INCLUDE VP;    ;                            VPX00030
                                                                      VPX00040
```

APPENDIX: LISTINGS OF MACRO FILES (ENTRY DECLARATIONS)

MACRO FILES:

CONS
EXECUTE
GET4
QEVENT
RTIMER
RTSTK
STIMER
SVCER
SVSTK
USERS
VPSTART

```
DCL CONS ENTRY (PTR, PTR) RETURNS (PTR);                    CON00010
```

CONVERSATIONAL MONITOR SYSTEM

FILE: EXECUTE PLI A

EXE00010

```
DCL EXECUTE ENTRY (CHAR(*) VAR);
```

```
DCL DSA FIXED BIN (31,0),                                       GET00010
    NAB FIXED BIN (31,0),                                       GET00020
    SEG FIXED BIN (31,0),                                       GET00030
    EOS FIXED BIN (31,0);                                       GET00040
                                                                GET00050
                                                                GET00060
DCL GET4 ENTRY (FIXED BIN (31), FIXED BIN (31),                 GET00070
                FIXED BIN (31), FIXED BIN (31))                 GET00080
        OPTIONS (ASSEMBLER, INTER);                             GET00090
```

```
DCL QEVENT ENTRY (FIXED BIN, FIXED BIN (31), CHAR(*) VAR,          QEV00010
                  FIXED BIN (31), PTR);                            QEV00020
```

CONVERSATIONAL MONITOR SYSTEM

FILE: RTIMER PLI A

RTI00010

```
DCL RTIMER ENTRY RETURNS (FIXED BIN (31));
```

```
DCL RTSTK ENTRY (PTR, FIXED BIN (31))                          RTS00010
        OPTIONS (ASSEMBLER, INTER);                            RTS00020
                                                               RTS00030
   /* ARG1: PTR -> SAVBLK                                      RTS00040
      ARG2: ADDR OF RETURN DSA (TOP DSA)                       RTS00050
   */                                                          RTS00060
                                                               RTS00070
```

```
DCL STIMER ENTRY RETURNS (FIXED BIN (31));                    STI00010
```

```
DCL SVCER ENTRY (FIXED BIN, PTR);                    SVC00010
```

```
                                                                SVS00010
DCL SVSTK ENTRY (PTR, FIXED BIN (31))                           SVS00020
        OPTIONS (ASSEMBLER, INTER);                             SVS00030
                                                                SVS00040
    /* ARG1: PTR -> SAVBLK                                      SVS00050
       ARG2: ADDR OF RETURN DSA (BOTTOM DSA)                    SVS00060
    */                                                          SVS00070
```

```
%INCLUDE STIMER;                                                    USE00010
DCL SEND ENTRY (FIXED BIN, FIXED BIN, FIXED BIN, CHAR(*) VAR,       USE00020
                FIXED BIN, PTR);                                    USE00030
DCL WAIT ENTRY (FIXED BIN);                                         USE00040
DCL SYNC ENTRY;                                                     USE00050
DCL FINISH ENTRY;                                                   USE00060
                                                                    USE00070
```

```
DCL VPSTART ENTRY (/* VP */, CHAR(*) VAR, PTR);                    VPS00010
```


END
DATE
FILMED
8 82
DTI